RZ/A1 Group

Graphics Library "RGA"

R01AN2162EJ0200
Rev. 2.00
Jul. 31, 2015

Introduction

This application note describes the Graphics Library RGA (Renesas Graphics Architecture) of RZ/A1.

The following lists features of the RGA.

- Allows high-speed drawing using the hardware acceleration.
- The API created based on the W3C standard HTML Canvas 2D Context achieves easy learning. In addition, the RGA provides C++API that is operable as an interface that is backward compatible with HTML Canvas 2D Context.
- The memory area provided by the application is available as a drawing destination or for input images.
- Allows drawing of translucent images and translucent drawing using an alpha mask.
- The RGA provides a conversion tool that can access image files as global variables. (This conversion tool operates on the host PC.)

Target Device

RZ/A1H, RZ/A1M, RZ/A1L

Restrictions

This library is not compatible with the vector graphics supported by Renesas Graphics Processor for OpenVG™ (R-GPVG2).

The hardware acceleration is used only for a part of drawing (Table 1-2), (6.11.1).

RZ/A1L does not use hardware acceleration. And it can not use a part of drawing. (Table 1-3)

This library cannot be used in multiple tasks. Use the RGA in a single task.

Contents

Graphics Library "RGA" 1

1. Specifications 4

2. File Configuration 9

3. Operation Verification Conditions 11

4. Reference Application Note(s) 12

5. Description of Hardware 13
5.1 Hardware Configuration 13
5.2 List of Pins to be Used 14

6. Description of Software 15
6.1 Outline of Operations 15
6.1.1 When Drawing on a Buffer Defined by the Application 15
6.1.2 Drawing on the Display Screen 17
6.2 Hardware resource allocation 20
6.3 Required Memory Size 21
6.4 Using OS resources 22
6.5 List of Constants 23
6.6 Types and Classes 24
6.6.1 Basic Types and Values 24
6.6.2 Error Codes 24
6.6.3 Types Only for the C Language 25
6.6.4 Classes Only for the C++ Language 27
6.6.5 Types/Classes Available for C Language and C++ Language 30
6.6.6 String Format 39
6.6.7 Type of porting layers 40
6.6.8 Changing state of class 41
6.7 List of Variables 42
6.8 Properties 43
6.8.1 Canvas2D_ContextClass Properties 43
6.8.2 Canvas2D_ImageClass Properties 45
6.9 Functions and Methods 46
6.9.1 Functions Equivalent to graphics_t Class Member Function 46
6.9.2 Functions Equivalent to graphics_image_t Class Member Function 64
6.9.3 Functions Equivalent to graphics_pattern_t Class Member Function 67
6.9.4 Functions Related to Creating C++ Object 68
6.9.5 Canvas2D_ContextClass Member Functions 68
6.9.6 Functions Related to Canvas2D_ImageClass 76
6.9.7 Functions Related to Canvas2D_PatternClass 76
6.9.8 Functions Related to WindowSurfacesClass 77
6.9.9 Functions Equivalent to window_surfaces_t Class Member Functions 80
6.9.10 Functions Related to byte_per_pixel_t Class 84
6.9.11 Functions Related to v_sync_t Class 85
6.9.12 Functions Related to vram_ex_stack_t class 88
6.9.13 Functions Related to animation_timing_function_t class 89
6.9.14 Other Functions 91
6.9.15 Functions in strings 93
6.10 Porting layer Functions 94
6.10.1 Functions on Default Settings of RGA 94
6.10.2 Porting layer functions of OSPL 96
6.10.3 Porting layer functions of RGPNCG 97
6.11 Supplementary Explanation 101
6.11.1 Correspondence to Canvas 2D and Correspondence to Hardware Acceleration 101
6.11.2 Identifying Image Format 104
6.11.3 Changing RGAH and RGAS 105

6.11.4 Flush Mode 106
6.11.5 Using SDRAM 107
6.11.6 Layer structure of sample screen controller 108
6.11.7 Flagged Structure Parameters 109
6.11.8 Defaultable Flags 110
6.11.9 Function to Initialize Internal Variables with Constants (*_initConst Function) 112
6.11.10 Finalize (*_Finalize Function) 113
6.11.11 Compatibility between C++ Language and JavaScript Object 115

7. Tools 116
7.1 Image Format Conversion by ImagePackager 116
7.1.1 Operational Procedure 116
7.1.2 List of files 116
7.1.3 Sample 116
7.1.4 Types of Output Binary File (Language) 117
7.1.5 File Formats in the Output Binary File 117
7.1.6 Input Formats 118
7.1.7 Parameters That Can Be Described in BinaryImageConfig.image.xml 118
7.2 Searching for Error Information by SearchErrorInformation 124
7.3 Converting binary by ConvertBin 125

8. Sample Codes 126

9. Reference Documents 127

Website and Support 128

Revision History 129

General Precautions in the Handling of MPU/MCU Products 130

Notice 131

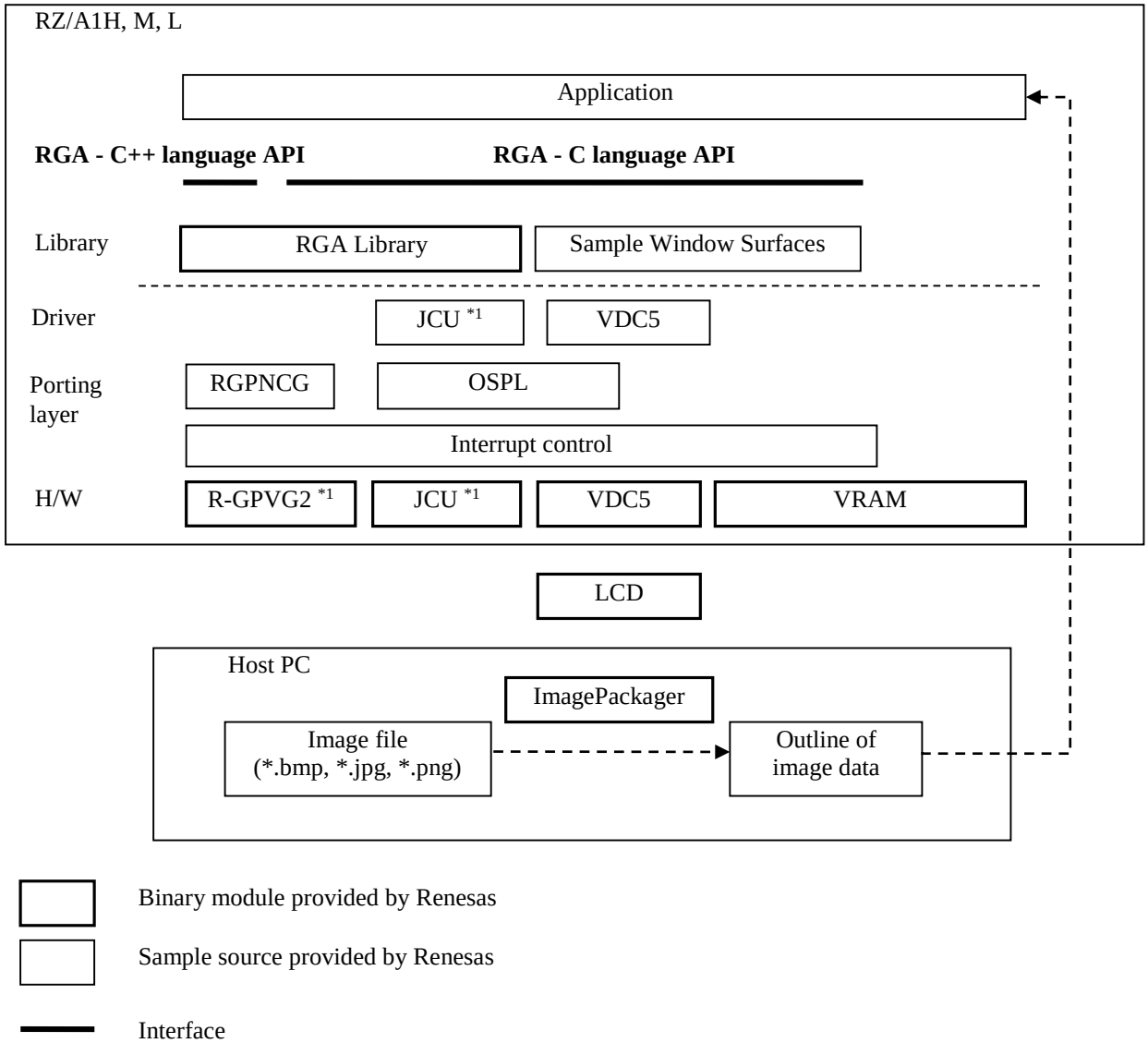
1. Specifications

The RGA is used to draw graphics images.

Table 1-1 lists Peripheral Functions to Be Used and Applications, Figure 1-1 shows a Block Diagram, and Table 1-2 and Table 1-3 list available pixel formats.

Table 1-1 Peripheral Functions to Be Used and Applications

Peripheral Function	Application
Renesas Graphics Processor for OpenVG™ (R-GPVG2)	Graphics drawing
JPEG Codec Unit (JCU)	JPEG decompression
Video Display Controller 5 (VDC5)	Screen display



*1 RZ/A1L does not have the function. The function is operated by software.

Figure 1-1 Block Diagram

The coordinate system of this library has an origin at the upper left. The value in the X-axis direction increases from left to right. The value in the Y-axis direction increases downward. The maximum width of a frame buffer to be drawn

is 1280 pixels, and its maximum height is 1024 pixels. The maximum size of source image is 2048x2048. First address and size of 1 line of the frame buffer and source image must be a multiple of 32.

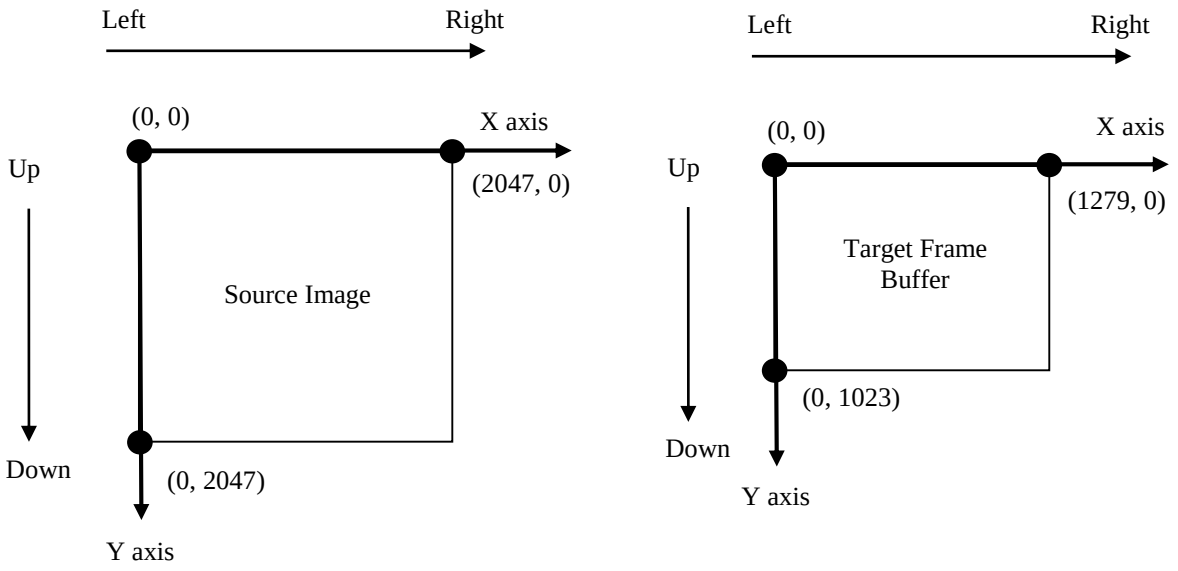


Figure 1-2 Max size of drawing target and source image

Table 1-2 Available Pixel Formats of Drawing Destination

	XRGB 8888	ARGB 8888[1]	RGB 565	ARGB 1555	ARGB 4444[1]	YCbCr 422	CLUT8	CLUT4	CLUT1
Hardware rendering	✓	✓	✓	✓	✓	x	x	x	x
Software rendering	✓	✓	✓	✓	✓	✓	✓	✓	✓
Matrix, enlargement/ reduction, blend	✓	✓	✓	✓	✓	x	x	x	x
Image drawing (6.9.1.29 etc.)	✓	✓	✓	✓	✓	✓	[2]	[2]	[2]
DrawImageChild (6.9.1.31)	✓	✓	✓	✓	✓	✓	x	x	x
Alpha only image	✓	✓	✓	✓	✓	x	x	x	x
Square fill	✓	✓	✓	✓	✓	✓	x	x	x
Byte per pixel (reference)	**4**	**4**	**2**	**2**	**2**	**2**	1	1/2	1/8

[1] RGB component is not multiplied alpha.

[2] Restrictions. For details, see the description on functions.

Table 1-3 Combinations of Available Pixel Formats of Images and Pixel Formats of Drawing Destination

Output / Input Image	XRGB 8888	ARGB 8888[1]	RGB 565	ARGB 1555	ARGB 4444[1]	YCbCr 422	CLUT8	CLUT4	CLUT1
JPEG	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	x	x	x
PNG	✓✓	✓✓	✓✓	✓✓	✓✓	x	x	x	x
XRGB8888	✓✓	✓	✓	✓	✓	x	x	x	x
ARGB8888	✓	✓✓	✓	✓	✓	x	x	x	x
RGB565	✓	✓	✓✓	✓	✓	x	x	x	x
ARGB1555	✓	✓	✓	✓✓	✓	x	x	x	x
ARGB4444	✓	✓	✓	✓	✓✓	x	x	x	x
R8G8B8A8[3]	✓✓	✓✓	✓✓	✓✓	✓✓	x	x	x	x
YUV422[2]	x	x	x	x	x	✓✓	x	x	x
CLUT8	x	x	x	x	x	x	✓✓	x	x
CLUT4	x	x	x	x	x	x	x	✓✓	x
CLUT1	x	x	x	x	x	x	x	x	✓✓
A8 [4]	✓	✓	✓	✓	✓	x	x	x	x
A4 [4]	✓	✓	✓	✓	✓	x	x	x	x
A1 [4]	✓	✓	✓	✓	✓	x	x	x	x

✓✓=RZ/A1H,M,L, ✓=RZ/A1H,M, x=Not available

The JPEG above is a case where JPEG data is specified for arguments of the image drawing function (R_GRAPHICS_DrawImage). Table 1-4 shows supported JPEG format. When a JPEG file or PNG file is converted to the raw format (including XRGB8888) by using the ImagePackager tool, see the column of the pixel format.

Table 1-4 Supported JPEG format

	RZ/A1H,M	RZ/A1L
Decoding module	JPEG Codec Unit (JCU) in RZ/A1	IJG JPEG (libjpeg)
JPEG standard	baseline	baseline
Pixel format in JPEG	YCbCr420 (H=2:1:1,V=2:1:1) YCbCr422 (H=2:1:1,V=1:1:1) YCbCr444 (H=1:1:1,V=1:1:1) YCbCr411 (H=4:1:1,V=1:1:1)	YCbCr420 (H=2:1:1,V=2:1:1) YCbCr422 (H=2:1:1,V=1:1:1)

Table 1-5 Supported PNG format

	RZ/A1H,M,L
Decoding module	LibPNG, zlib

[3] If the source image was R8G8B8A8 or YCbCr422, S/W rendering is used.

[4] Width of A8, A4 and A1 format image must be same as width of drawing target frame buffer.

The following tables describes the detail of pixel format. RGA for RZ/A1 is little endian. For example, Red of XRGB8888 is at first address of the pixel + 2.

XRGB8888

bit 31 – 24	23 – 16	15 – 8	7 – 0
0	Red	Green	Blue

ARGB8888

bit 31 – 24	23 – 16	15 – 8	7 – 0
Alpha	Red	Green	Blue

RGB565

bit 15 – 11	10 – 5	4 – 0
Red	Green	Blue

ARGB1555

bit 15	14 – 10	9 – 5	4 – 0
Alpha	Red	Green	Blue

ARGB4444

bit 15 – 12	11 – 8	7 – 4	3 – 0
Alpha	Red	Green	Blue

R8G8B8A8

address +0	+1	+2	+3
Red	Green	Blue	Alpha

YCbCr422

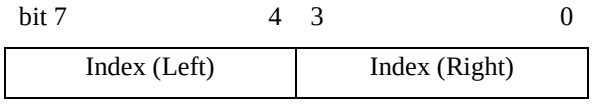
address +0	+1	+2	+3
Cb	Y (Left)	Cr	Y (Right)

FourCC = UYVY, CbCr center (grayscale) = 0x80

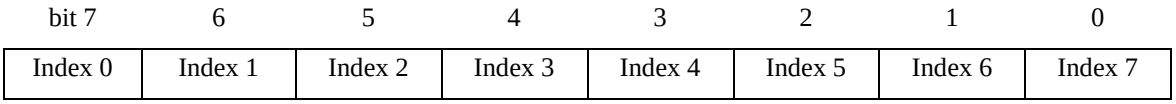
CLUT8

bit 7 – 0
Index

CLUT4

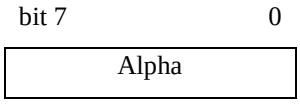


CLUT1

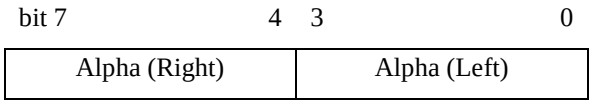


Index 0 is the most left pixel. Index 7 is the most right pixel.

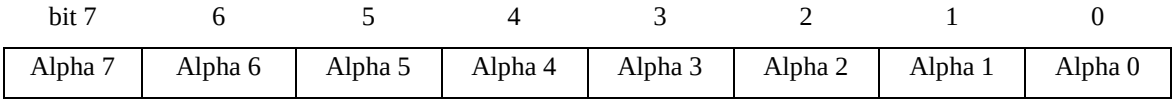
A8



A4

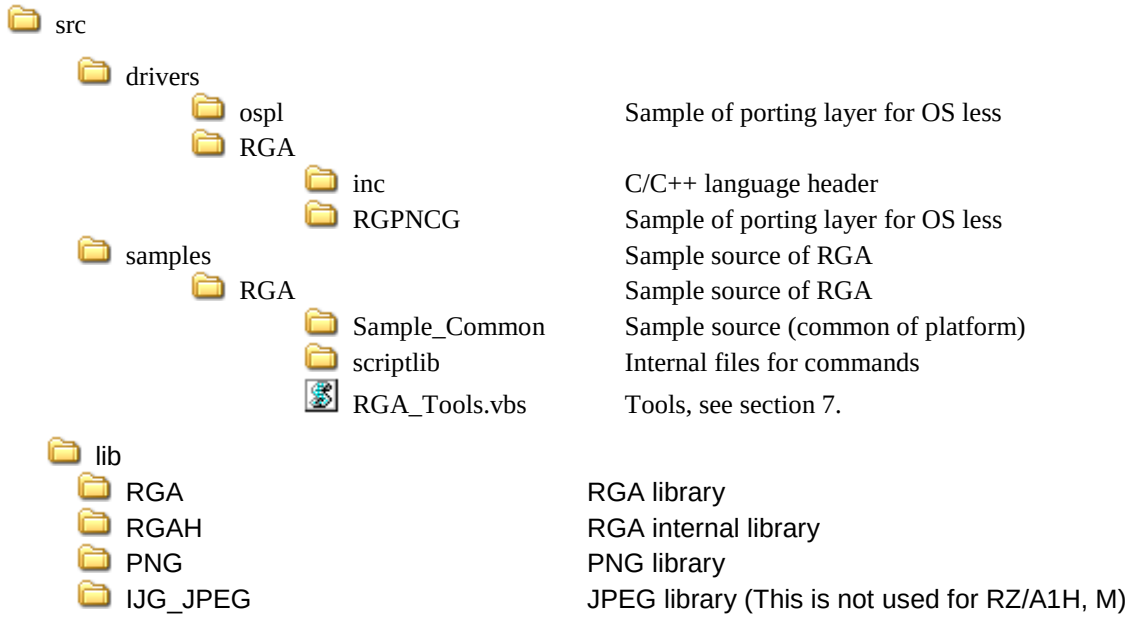


A1



Alpha 0 is the most left pixel. Alpha 7 is the most right pixel.

2. File Configuration



When you installed RGA library to your project folder, copy "src\drivers\inc" folder and "lib" folder. When link error was raised, copy source files defined error raised function. When an error of not found #include file, copy header files. And copy source files of drivers and common functions defined functions that was raised the unreferenced link error.

Write '#include "RGA.h"' in the application program using RGA.

Set the work buffer of RGA in uncached area and put in available memory area. The address of work buffer (the value of "work_buffer_address" member variable in "R_GRAPHICS_STATIC_OnInitializeDefault" function) is changed to uncached area by calling "R_OSPL_ToUncachedAddress" function in "r_ospl_memory.c" file from inside of RGA (or checked to be already in uncached area) and changed to physical address accessible from hardware by calling "R_OSPL_ToPhysicalAddress" function.

If memory area became few, reduce frame buffer to the size of showing only and set the size of work buffer B to 0.

Check that the memory map by setting in "r_ospl_memory.c" is same as the setting of MMU. See "Memory" section in "OS porting guide" section in application note of RZ/A1H group OS porting layer "OSPL" (R01AN1887JJ).

Table 2-1 Main header files (in src\drivers\RGA\inc)

File Name	Description
RGA.h	Main of RGA
RGA_API.h	Sub of RGA : API. This file is included from RGA.h
RGA_Config.h	Sub of RGA : setting. This file is included from RGA.h
RGA_Cpp.h	Sub of RGA : C++ API. This file is included from RGA.h

Table 2-2 Library files

File Name	RGA hardware rendering	RGA software rendering	Description
libRGA.a	✓	✓	Main of RGA Hardware accelerator using version is in *ForRGAH* folder. No hardware accelerator using version is *ForRGAS* folder. libRGA.a has the rendering routine (RGAS).
libRGAH.a	✓	-	libRGA.a internal module (RGAH). This is not used by RGA for no hardware accelerator version.

libPNG.a	✓	✓	This decodes PNG image. Both RGA for hardware accelerator version and no hardware accelerator version use *libPNG.a*. Both RGA do not use hardware accelerator for decoding PNG image. The environment of gcc not supported longjmp function can not continue after raising an error. It calls *user_error_fn* function and *PNG_ABORT* macro. Set the size of work buffer for PNG module to *work_size_for_libPNG* member variable. This library can be made from source files in *PNG* folder. If you want to update the newest LibPNG module, modify and build source files for RZ/A1 by source files for reference.
libjpeg.a	-	✓	This decodes JPEG image. RGA for hardware accelerator version does not use libjpeg.a. This library can be made from source files in *IJG_JPEG* folder. If you want to update the newest libjpeg module, modify and build source files for RZ/A1 by source files for reference.

3. Operation Verification Conditions

Operations of the sample code of this application note are verified under the following conditions.

Table 3.1 Operation Verification Conditions

Item	Description
MCU used	RZ/A1
Operating frequency	CPU clock (Iφ): 400MHz Image processing clock (Gφ): 266.67MHz Internal bus clock (Bφ): 133.33MHz Peripheral clock 1 (P1φ): 66.67MHz Peripheral clock 0 (P0φ): 33.33MHz
Operating voltage	Power supply voltage (I/O): 3.3V Power supply voltage (Internal): 1.18V
Integrated development environment	ARM® integrated development environment ARM Development Studio 5 (DS-5TM) Version 5.16
C compiler	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
	Compiler options (Excluded directory path) -O3 -Ospace --cpu=Cortex-A9 --littleend --arm --apcs=/interwork --no_unaligned_access --fpu=vfpv3_fp16 -g
Operating mode	Boot mode 0 (CS0 space 16bit boot)
Communication setting of terminal software	• Communication speed: 115200bps • Data length: 8 bits • Parity: None • Stop bit length: 1 bit • Flow control: None
Board used	GENMAI board • RTK772100BC00000BR (R7S72100 CPU board) • RTK77210000B00000BR (R7S72100 Option board)
Device used	LCD. Serial interface (D-sub 9-pin connector)

4. Reference Application Note(s)

For additional information associated with this document, refer to the following application note(s).

- RZ/A1H Group Example of Initialization (R01AN1646EJ)
- RZ/A1H Group I/O definition header file <iodefine.h> (R01AN1860EJ)
- RZ/A1H Group OS porting layer "OSPL" Sample Program (R01AN1887EJ)

5. Description of Hardware

5.1 Hardware Configuration

Figure 5.1 shows Example of Hardware Configuration.

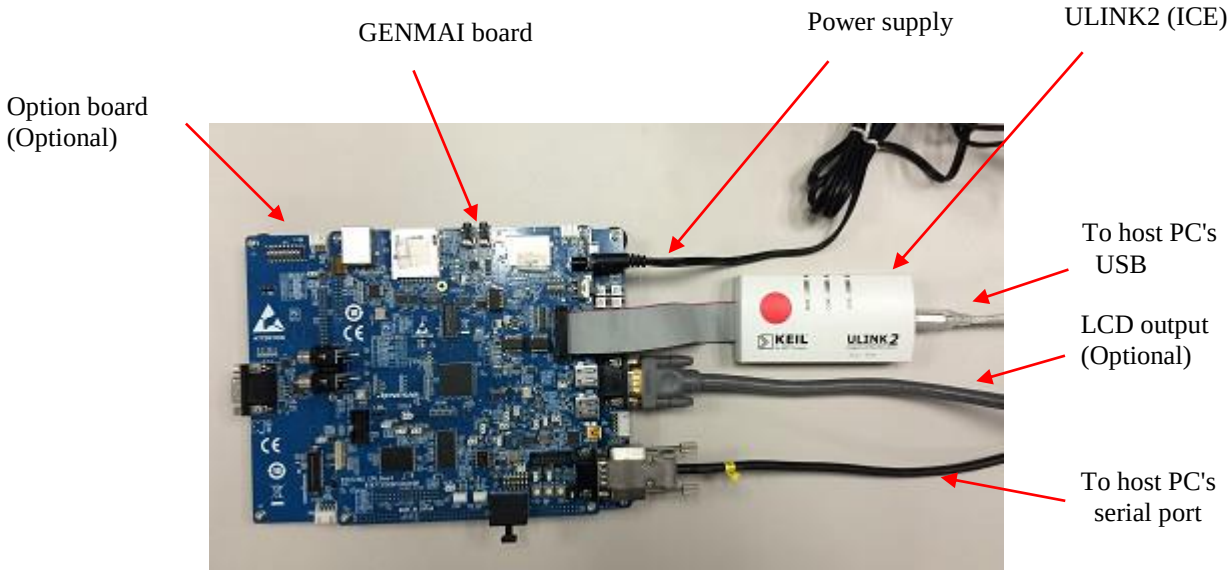


Figure 5.1 Example of Hardware Configuration (RZ/A1H)

5.2 List of Pins to be Used

Table 5-1 lists the pins to be used and their functions.

Table 5-1 Pins to be Used and their Functions

Pin name	I/O	Description
None		

6. Description of Software

6.1 Outline of Operations

6.1.1 When Drawing on a Buffer Defined by the Application

6.1.1.1 Flowchart

Figure 6.1 shows a flowchart of drawing on a frame buffer defined by the application.

For the actual operation procedure, see the attached document, RGA Tutorial.

When using the C++ language API, see the description below using the C language API and the relationship between the C language API described in section 6.11.1, Correspondence to Canvas 2D and Correspondence to Hardware Acceleration, and the C++ language API.

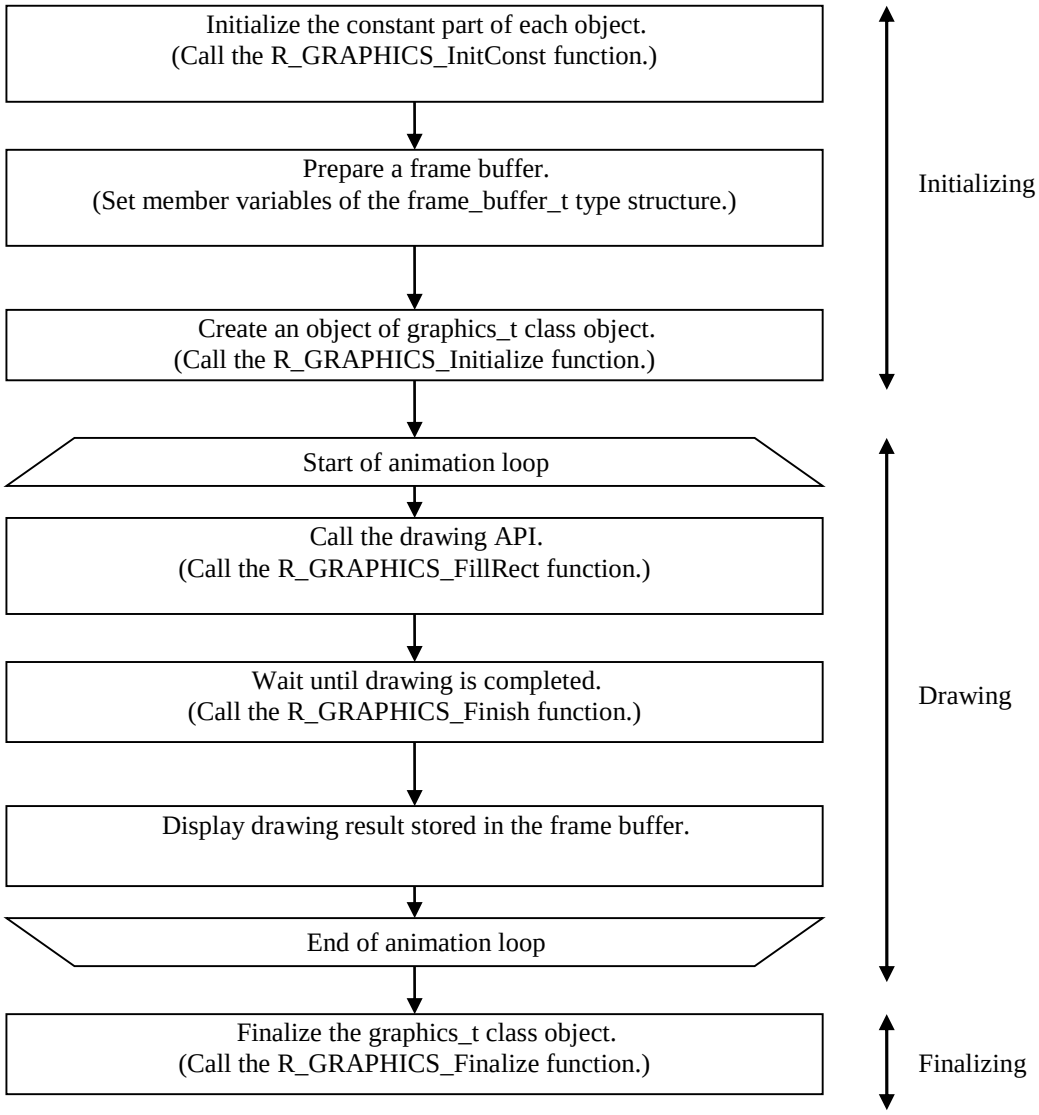

Initialize the constant part of each object.
(Call the R_GRAPHICS_InitConst function.)
Prepare a frame buffer.
(Set member variables of the frame_buffer_t type structure.)
Create an object of graphics_t class object.
(Call the R_GRAPHICS_Initialize function.)
Initializing
Start of animation loop
Call the drawing API.
(Call the R_GRAPHICS_FillRect function.)
Wait until drawing is completed.
(Call the R_GRAPHICS_Finish function.)
Display drawing result stored in the frame buffer.
End of animation loop
Drawing
Finalize the graphics_t class object.
(Call the R_GRAPHICS_Finalize function.)
Finalizing

Figure 6.1 Drawing on a Buffer Defined by the Application - Flowchart

6.1.1.2 Sequence Chart

The following shows an operation timing chart of the software (application and library) and hardware (drawing hardware and display hardware) in the case of drawing on the frame buffer provided by the application.

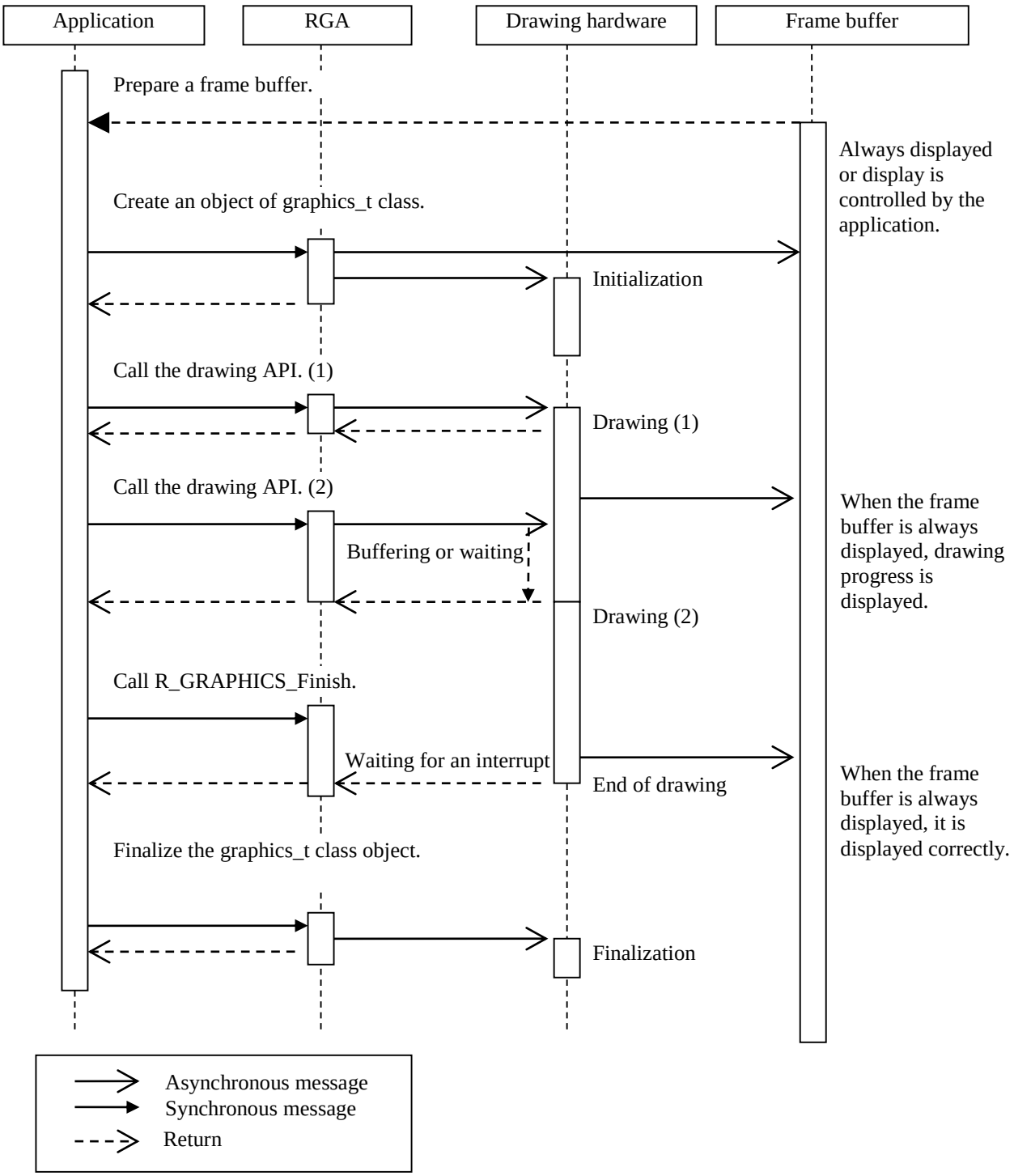


Figure 6.2 Operation sequence

6.1.2 Drawing on the Display Screen

6.1.2.1 Flowchart

Figure 6.3 shows a flowchart of drawing on the display screen.

For the actual operation procedure, see the attached document, RGA Tutorial.

When using the C++ language API, see the description below using the C language API and the relationship between the C language API described in section 6.11.1, Correspondence to Canvas 2D and Correspondence to Hardware Acceleration, and the C++ language API.

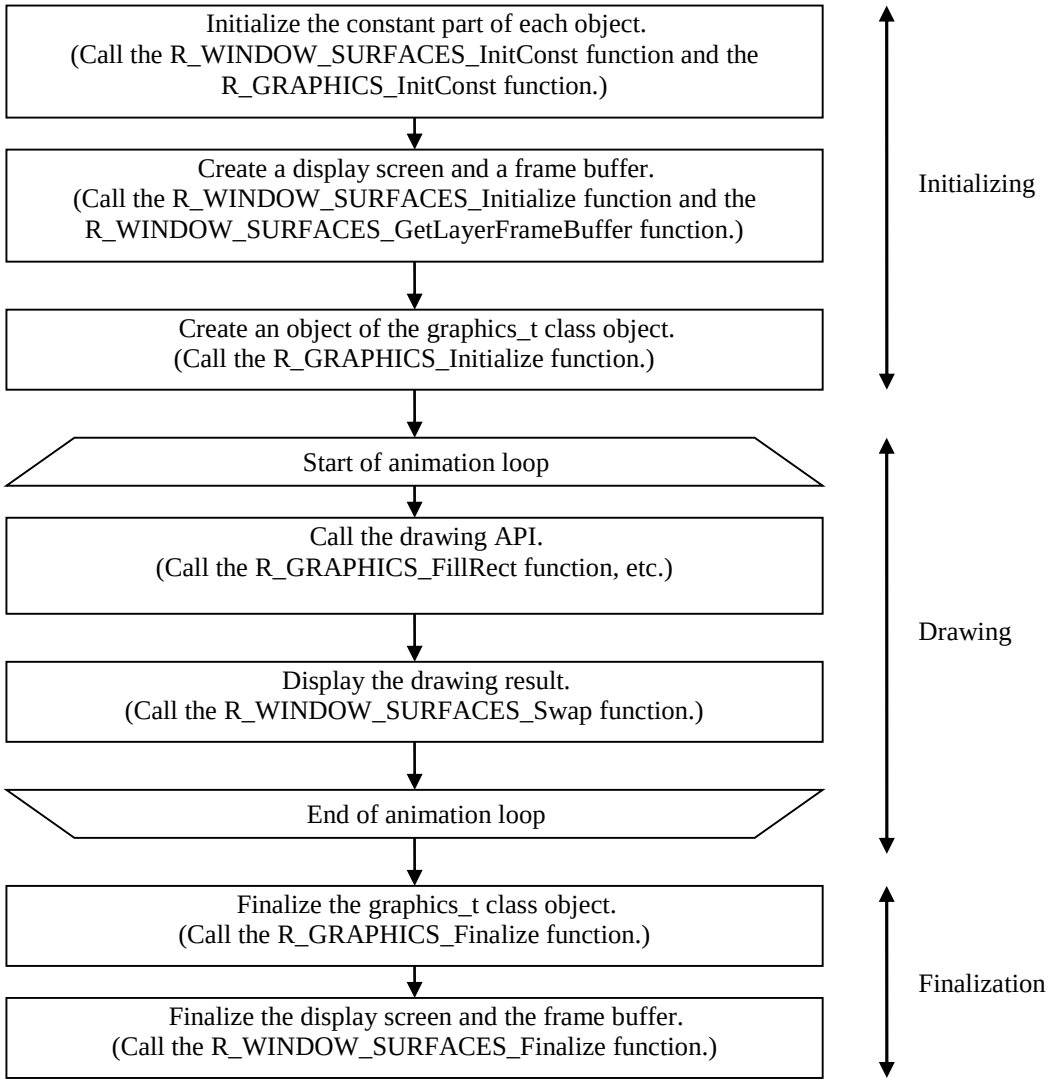

Initialize the constant part of each object.
(Call the R_WINDOW_SURFACES_InitConst function and the R_GRAPHICS_InitConst function.)
Create a display screen and a frame buffer.
(Call the R_WINDOW_SURFACES_Initialize function and the R_WINDOW_SURFACES_GetLayerFrameBuffer function.)
Create an object of the graphics_t class object.
(Call the R_GRAPHICS_Initialize function.)
Start of animation loop
Call the drawing API.
(Call the R_GRAPHICS_FillRect function, etc.)
Display the drawing result.
(Call the R_WINDOW_SURFACES_Swap function.)
End of animation loop
Finalize the graphics_t class object.
(Call the R_GRAPHICS_Finalize function.)
Finalize the display screen and the frame buffer.
(Call the R_WINDOW_SURFACES_Finalize function.)
Initializing
Drawing
Finalization

Figure 6.3 Drawing on the Display Screen - Flowchart Processing

6.1.2.2 Sequence Chart

The following shows an operation timing chart of the software (application and library) and hardware (drawing hardware and display hardware) in the case of drawing on the frame buffer provided by the RGA's WindowSurfaces library.

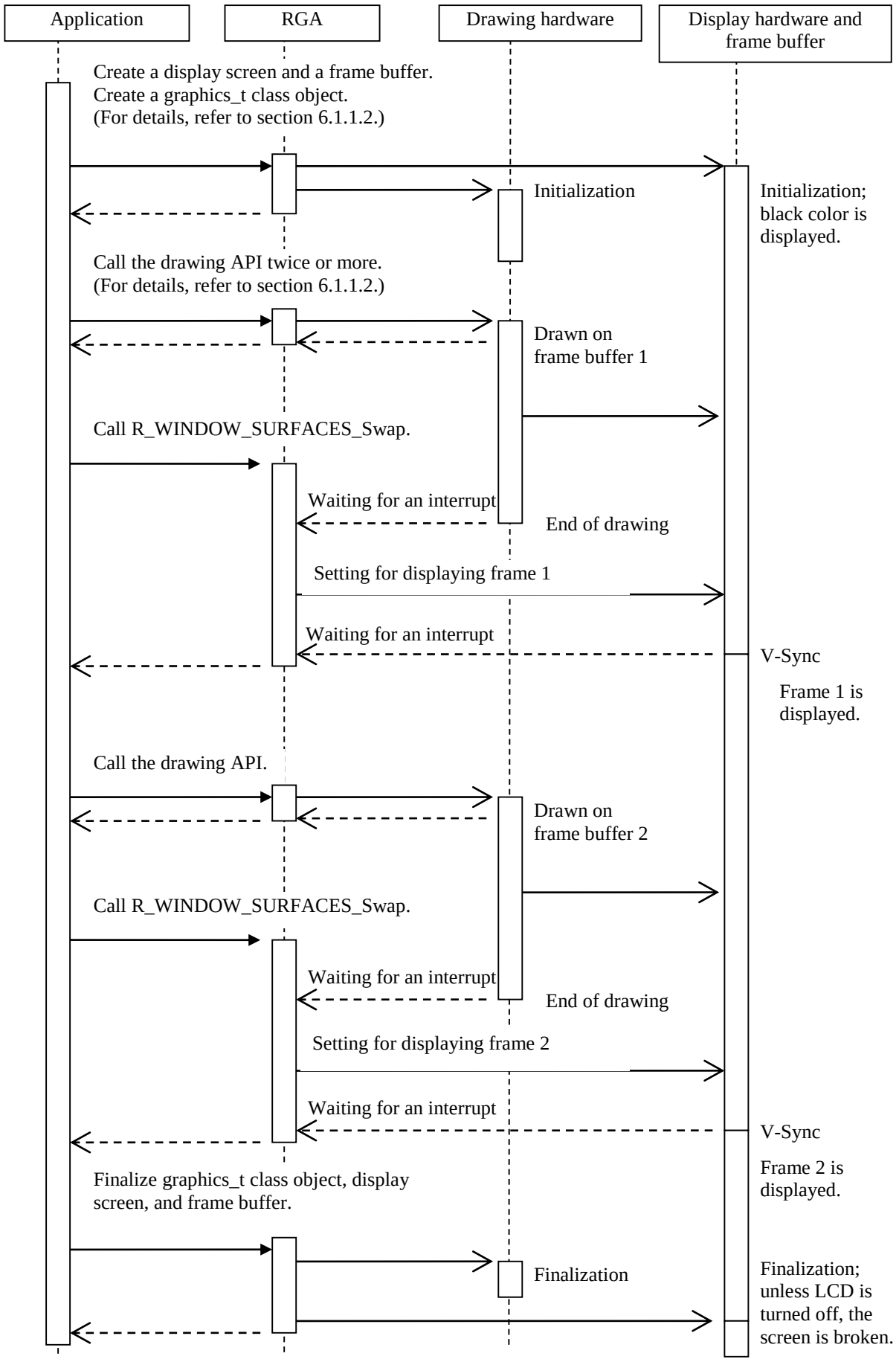

Application
RGA
Drawing hardware
Display hardware and frame buffer
Create a display screen and a frame buffer.
Create a graphics_t class object.
(For details, refer to section 6.1.1.2.)
Initialization
Initialization; black color is displayed.
Call the drawing API twice or more.
(For details, refer to section 6.1.1.2.)
Drawn on frame buffer 1
Call R_WINDOW_SURFACES_Swap.
Waiting for an interrupt
End of drawing
Setting for displaying frame 1
Waiting for an interrupt
V-Sync
Frame 1 is displayed.
Call the drawing API.
Drawn on frame buffer 2
Call R_WINDOW_SURFACES_Swap.
Waiting for an interrupt
End of drawing
Setting for displaying frame 2
Waiting for an interrupt
V-Sync
Frame 2 is displayed.
Finalize graphics_t class object, display screen, and frame buffer.
Finalization
Finalization; unless LCD is turned off, the screen is broken.

Figure 6.4 Operation sequence

6.2 Hardware resource allocation

Renesas Graphics Processor for OpenVG™ (R-GPVG2) and JPEG Codec Unit (JCU) are allocated by an instance as "graphics_t" class. If 2nd instance as "graphics_t" class was initialized, an error is raised.

Video Display Controller 5 (VDC5)'s hardware layers are allocated with layers in "window_surfaces_t" object.

If "graphics_config_t ::jpeg_decoder = GRAPHICS_JPEG_DECODER_NONE", RGA does not access to JPEG Codec Unit (JCU).

6.3 Required Memory Size

Table 6.1 shows Required Memory Size.

Table 6.1 Required Memory Size

Memory	Size (bytes)	Note
ROM	32732	libRGA.a - Code
ROM	25240	libRGAH.a - Code
ROM	48	libRGA.a - Read Only Data
ROM	0	libRGAH.a - Read Only Data
ROM, RAM	802	libRGA.a - Read Write Data
ROM, RAM	28	libRGAH.a - Read Write Data
RAM	0	libRGA.a - Zero Initialized Data
RAM	940	libRGAH.a - Zero Initialized Data
RAM	Refer to R_RGA_CalcWorkBufferSize 6.9.14.3	Work buffer
RAM	Refer to R_RGA_CalcWorkBufferB_Size 6.9.14.4	Work buffer B
RAM	1536000 (2x16bit x800x480)	Frame buffer
RAM	2304000 (3x16bit x800x480)	Back buffer for application

Note: These required memory sizes vary depending on the C compiler version and the compile option.

6.4 Using OS resources

RGPNCG attached RGA for RTX uses the thread assigned signal in "NCGSYS_*_State" function. All bits of event will be used.

6.5 List of Constants

Table 6.2 lists Constants Used for the Sample Code.

Table 6.2 Constants Used for the Sample Code

Constant Name	Value for RGAH	Value for RGAS	Description
RGA_VERSION		120	RGA version
RGA_FRAME_BUFFER_ADDRESS_ALIGNMENT	32	4	Address alignment of drawing target frame buffer (bytes)
RGA_SOURCE_IMAGE_STRIDE_ALIGNMENT	32	4	Byte count alignment to next line (stride) of source image (bytes)
RGA_DESTINATION_STRIDE_ALIGNMENT	32	4	Byte count alignment to next line (stride) of drawing target (bytes)
RGA_JPEG_ADDRESS_ALIGNMENT	8	4	Address alignment of JPEG data (bytes)
RGA_JPEG_MAX_WIDTH_ALIGNMENT		16	Maximum width of MCU of source JPEG image. (pixel)
RGA_JPEG_MAX_HEIGHT_ALIGNMENT		16	Maximum height of MCU of source JPEG image. (pixel)
RGA_VDC5_BUFFER_ADDRESS_ALIGNMENT		64	First address alignment of the frame buffer shown by Video Display Controller 5 (VDC5) (bytes)
RGA_WORK_BUFFER_STRIDE	64	-	One line size of work buffer (bytes)
RGA_WORK_BUFFER_ADDRESS_ALIGNMENT	64	4	First address alignment of work buffer (bytes)
RGA_WORK_BUFFER_HEIGHT_ALIGNMENT	8	-	Height alignment of work buffer (pixel)
RGA_WORK_BUFFER_B_ADDRESS_ALIGNMENT	32	4	First address alignment of work buffer B (bytes)
RGA_MAX_WORK_SIZE_FOR_LIB_PNG_DEFAULT		0x40000	Default size (bytes) of memory area as heap area allocated by PNG decoding. This memory area is allocated in work buffer B See work_size_for_libPNG member variable.

6.6 Types and Classes

6.6.1 Basic Types and Values

Symbol	Description
char_t	8-bit character
bool_t	Logical data type. The value is true (1) or false (0).
int_t	The signed integer for this library is a 32-bit signed integer.
int8_t	8-bit signed integer (defined by standard library)
int16_t	16-bit signed integer (defined by standard library)
int32_t	32-bit signed integer (defined by standard library)
int64_t	64-bit signed integer (defined by standard library)
uint8_t	8-bit unsigned integer (defined by standard library)
uint16_t	16-bit unsigned integer (defined by standard library)
uint32_t	32-bit unsigned integer (defined by standard library)
uint64_t	64-bit unsigned integer (defined by standard library)
int_fast8_t	Fastest 8-bit minimum-width signed integer
int_fast16_t	Fastest 16-bit minimum-width signed integer
int_fast32_t	Fastest 32-bit minimum-width signed integer
uint_fast8_t	Fastest 8-bit minimum-width unsigned integer
uint_fast16_t	Fastest 16-bit minimum-width unsigned integer
uint_fast32_t	Fastest 32-bit minimum-width unsigned integer
uintptr_t	Same as pointer bit width unsigned integer as physical address
size_t	Same as pointer bit width unsigned integer as byte size
ptrdiff_t	Same as pointer bit width signed integer as difference between pointers
bit_flags_fast32_t	Same as uint_fast32_t bit flags (bit field)
bit_flags32_t	Same as uint32_t bit flags (bit field)
float32_t	32-bit float (Defined by standard library when "__ARM_NEON__" defined)
float64_t	64-bit float (Defined by standard library when "__ARM_NEON__" defined)
float128_t	128-bit float

6.6.2 Error Codes

Error code can be got from the return value of C language API. Error code of C++ API can be got from "R_OSPL_GetErrNum" function.

Refer to (7.2) Searching for Error Information by SearchErrorInformation.

Symbol	Value	Description
0	0	No error
E_OTHERS	1	Others error
E_FEW_ARRAY	2	Error of few fixed length array
E_FEW_MEMORY	3	Few heap memory area
E_FIFO_OVER	4	Failed to enqueue
E_NOT_FOUND_SYMBOL	5	Not defined the symbol
E_NO_NEXT	6	There is not next element of list
E_ACCESS_DENIED	7	Error of denied read or write
E_NOT_IMPLEMENT_YET	9	Not implemented yet
E_ERRNO	0x0E(=14)	Refer to "errno"
E_LIMITATION	0x0F(=15)	Limitation
E_STATE	0x10(=16)	Cannot do at this state
E_PATH_NOT_FOUND	0x12(=18)	Not found file or folder
E_BAD_COMMAND_ID	0x16(=22)	Out of number of command ID
E_TIME_OUT	0x17(=23)	Time out

E_NO_DEBUG_TLS	0x1D(=29)	Not set debug work area. Refer to "R_OSPL_SET_DEBUG_WORK" [5]
E_EXIT_TEST	0x1E(=30)	Request of exit from the test
E_NOT_SUPPORTED_PIXEL_FORMAT	0x9400 =37888	Unsupported pixel format. Change to supported pixel format. See Table 1-2 Available Pixel Formats of Drawing Destination

6.6.3 Types Only for the C Language

The following are types supplied by the RGA and used only for the C language.

Type definitions shown in this section are also available for the C++ language, but there are classes available for the C++ language. For details, see section 6.6.4.

For types available for the C language, also see section 6.6.5, Types/Classes Available for C Language and C++ Language.

6.6.3.1 List of Types

Table 6.3 C Language-Dedicated Types Provided by RGA

Section	Type	Description
6.6.3.2	graphics_t	Graphics drawing context
6.6.3.3	graphics_image_t	Input image
6.6.3.4	graphics_pattern_t	Image arranged pattern

6.6.3.2 graphics_t

graphics_t structure	
Outline	Graphics drawing context
Header	RGA.h
Description	Function equivalent to the member function: See section 6.9.1. Renesas Graphics Processor for OpenVG™ (R-GPVG2) or software rendering is used in the RZ/A1H.
Member Variable	Access Inhibit

6.6.3.3 graphics_image_t

graphics_image_t structure	
Outline	Image
Header	RGA.h
Description	Function equivalent to the member function: See section 6.9.1.42. The ImagePackager tool converts this type from an image file and outputs this type data. It is also possible to dynamically generate a graphics_image_t-class object without using the ImagePackager tool.
Member Variable	Access Inhibit

6.6.3.4 graphics_pattern_t

graphics_pattern_t structure	
Outline	image-arranged pattern
Header	RGA.h

[5] RZ/A1H group OS Porting Layer "OSPL" Sample Program (R01AN1887EJ)

Description	Function equivalent to the member function: See section 6.9.3.
Member Variable	Access Inhibit

6.6.4 Classes Only for the C++ Language

The following describes C++ language classes provided by the RGA.

Be careful of coding because classes provided by the RGA are created to be compatible with JavaScript. For details, see section 6.11.11 Compatibility between C++ Language and JavaScript Object.

For classes/types available for the C language, also see section 6.6.5, Types/Classes Available for C Language and C++ Language.

6.6.4.1 List of Types

Table 6.4 C++ Language-Dedicated Types Provided by RGA

Section	Type	Description
6.6.4.2	Canvas2D_ContextClass	Graphics drawing context
6.6.4.3	Canvas2D_ContextConfigClass	Setting of "Canvas2D_ContextClass"
6.6.4.4	Canvas2D_ImageClass	Input image
6.6.4.5	Canvas2D_PatternClass	Image-arranged pattern
6.6.4.6	WindowSurfacesClass	Frame buffer and showing screen
6.6.4.7	WindowSurfacesConfigClass	Argument of "WindowSurfacesClass::initialize"
6.6.4.8	LayerAttributesClass	Argument of "WindowSurfacesClass::access_layer_attributes"

6.6.4.2 Canvas2D_ContextClass

```
#include  "RGA.h"
class  Canvas2D_ContextClass;
```

This is a graphics drawing context class only for the C++ language. This class follows a coding rule of JavaScript.

Renesas Graphics Processor for OpenVG™ (R-GPVG2) or software rendering is used in the RZ/A1H.

For the description on properties, refer to property specifications (section 6.8.1).

For the description on member functions, refer to function specifications (sections 6.9.5 and 6.9.4.2).

The graphics_t type member functions are made available by using the c_LanguageContext property. See section 6.9.1.

Sample:

```
Canvas2D_ContextClass  canvas2d = R_RGA_New_Canvas2D_ContextClass( frame );

/* JavaScript start of compatible part */
canvas2d.clearRect( 0, 0, frame_width, frame_height );
canvas2d.Style = "#0f0";
canvas2d.fillRect( 100, 100, 200, 100 );
/* JavaScript end of compatible part */

if ( R_OSPL_GetErrNum() != 0 ) { /* error */ }

R_WINDOW_SURFACES_SwapBuffers( ... );
canvas2d.destroy();
```

6.6.4.3 Canvas2D_ContextConfigClass

Canvas2D_ContextConfigClass structure	
Outline	This is a setting of "Canvas2D_ContextClass" for the C++ language

Header RGA.h
Description Default value is set by the constructor.

Member Variable

Symbol	Description
frame_buffer_t* frame_buffer	Drawing target frame buffer. This variables must be set. Set the address of frame_buffer_t type structure.
bool_t is_fast_manual_flush	Fast manual flush mode (6.11.4) or not. Default value is "false".

6.6.4.4 Canvas2D_ImageClass

```
#include "RGA.h"
class Canvas2D_ ImageClass;
```

This is a referable image object type only for the C++ language. This class follows a coding rule of JavaScript.

For the description on properties, refer to property specifications (section 6.8.2).

For the description on member functions, refer to function specifications (section 6.9.6).

6.6.4.5 Canvas2D_PatternClass

```
#include "RGA.h"
class Canvas2D_PatternClass;
```

This is an image-arranged pattern class only for the C++ language. This class follows a coding rule of JavaScript.

For the description on member functions, refer to function specifications (section 6.9.7).

6.6.4.6 WindowSurfacesClass

```
#include "RGA.h"
class WindowSurfacesClass;
```

This is a frame buffer and showing screen class only for the C++ language. This class follows a coding rule of mbed.

For the description on member functions, refer to function specifications (section 6.9.8).

6.6.4.7 WindowSurfacesConfigClass

```
#include "RGA.h"
class WindowSurfacesConfigClass;
```

This is a class of argument for "WindowSurfacesClass::initialize" member function for C++ language. This class follows a coding rule of mbed.

Member variables of this class are inherited from "window_surfaces_config_t" type. See 6.6.5.6. Also, member variables are initialized to default value by constructor. It is not necessary to set "flags" member variable.

6.6.4.8 LayerAttributesClass

```
#include "RGA.h"
class LayerAttributesClass;
```

This is a class of argument for "WindowSurfacesClass::access_layer_attributes" member function for C++ language. This class follows a coding rule of mbed.

Member variables of this class are inherited from "layer_attributes_t" type. See 6.6.5.7. Also, member variables are initialized to default value by constructor. It is not necessary to set "flags" member variable.

6.6.5 Types/Classes Available for C Language and C++ Language

For types available from the C language, also see section 6.6.3, Types Only for the C Language.

For types available from the C++ language, also see section 6.6.4, Classes Only for the C++ Language.

6.6.5.1 List of Classes/Types

Table 6.5 Types/Classes Provided by RGA, which are Available for C Language and C++ Language

Section	Type	Description
6.6.5.2	frame_buffer_t	Drawing destination frame buffer
6.6.5.3	graphics_config_t	Type for setting graphics_t
6.6.5.4	graphics_quality_flags_t	Defaultable flag indicating drawing quality
6.6.5.5	window_surfaces_t	Frame buffer and screen display sample
6.6.5.6	window_surfaces_config_t	window_surfaces_t setting
6.6.5.7	layer_attributes_t	Parameter of the R_WINDOW_SURFACES_AccessLayerAttributes function
6.6.5.8	access_t	Operations such as specifying set values (write) and acquiring set values (read)
6.6.5.9	byte_per_pixel_t	Number of bytes per pixel
6.6.5.10	pixel_format_t	Pixel format
6.6.5.11	frame_buffer_delegate_t	Data related to frame buffer
6.6.5.12	v_sync_t	V-Sync signal synchronization of display screen
6.6.5.13	vram_ex_stack_t	Stack for off-screen buffer in external RAM
6.6.5.14	graphics_image_properties_t	Properties of the image
6.6.5.15	graphics_composite_operation_t	Type of composite operation
6.6.5.16	graphics_status_t	Area to store graphics drawing context
6.6.5.17	graphics_matrix_float_t	Element of matrix
6.6.5.18	repetition_t	Pattern repetition type
6.6.5.19	r8g8b8a8_t	Pixel format arranged in the order of R, G, B, and A
6.6.5.20	animation_timing_function_t	Timing of animation (Bezier function)
6.6.5.21	graphics_jpeg_decoder_t	Constant value of a kind of JPEG decoder
6.6.5.22	graphics_async_status_t	Status of drawing graphics and interrupt status

6.6.5.2 frame_buffer_t

frame_buffer_t structure

Outline: Drawing destination frame buffer type

Header: RGA.h, frame_buffer.h

Description

Member Variable

Symbol	Description
uint8_t* buffer_address[ ]	Array of starting address of the frame buffer, logical address of cache area (same as normal variables). Specify a multiple of 32.
int_fast32_t buffer_count	Number of buffer_address array elements (1: Single buffer, 2: Double buffers)
int_fast32_t show_buffer_index	Buffer number being displayed or used as source image (buffer_address array number)
int_fast32_t draw_buffer_index	Buffer number being drawn (buffer_address array number)
int_fast32_t width	Frame buffer width (pixels)
byte_per_pixel_t byte_per_pixel	Number of bytes per pixel. See Table 1-2

int_fast32_t stride	Number of bytes of pixels having the same x coordinate in the previously below line. Specify a multiple of 32.
int_fast32_t height	Frame buffer height (pixels)
pixel_format_t pixel_format	Pixel format
frame_buffer_delegate_t* delegate	User-defined variable

Sample:

```
static uint8_t  gs_frame_buffer_memory[2][800][480][4];
frame_buffer_t  frame;
frame.buffer_address[0] = gs_frame_buffer_memory [0];
frame.buffer_address[1] = gs_frame_buffer_memory [1];
frame.buffer_count = 2;
frame.show_buffer_index = 0;
frame.draw_buffer_index = 1;
frame.width = 800;
frame.byte_per_pixel = 4;
frame.stride = 800 * 4;
frame.height = 480
frame.pixel_format = PIXEL_FORMAT_ARGB8888;
frame.delegate = NULL;
```

6.6.5.3 graphics_config_t

graphics_config_t structure

Outline: Configuration of graphics_t type

Header: RGA.h

Description: To customize default values or assemble essential settings, use the function described in section 6.10.1.3, R_GRAPHICS_OnInitialize_FuncType.

Member Variable

Symbol	Description
bit_flags_fast32_t flags	See section 6.11.7, Flagged Structure Parameters. (mandatory) F_GRAPHICS_FRAME_BUFFER F_GRAPHICS_WORK_BUFFER_ADDRESS F_GRAPHICS_WORK_BUFFER_SIZE F_GRAPHICS_MAX_HEIGHT_OF_FRAME_BUFFER F_GRAPHICS_QUALITY_FLAGS F_GRAPHICS_BACKGROUND_COLOR F_GRAPHICS_IS_FAST_MANUAL_FLUSH F_GRAPHICS_WORK_BUFFER_B_ADDRESS F_GRAPHICS_WORK_BUFFER_B_SIZE F_GRAPHICS_JPEG_DECODER F_GRAPHICS_WORK_SIZE_FOR_LIBPNG F_GRAPHICS_INTERNAL_EVENT_VALUE F_GRAPHICS_LOCK_OBJECT F_GRAPHICS_CHANNEL_OF_DMAC_1 F_GRAPHICS_CHANNEL_OF_DMAC_2 F_GRAPHICS_EVENT_VALUE_OF_DMAC_1 F_GRAPHICS_EVENT_VALUE_OF_DMAC_2
frame_buffer_t* frame_buffer	Frame buffer to be drawn for which member variables must be set. Set the address of frame_buffer_t type structure.
void* work_buffer_address	Work buffer starting address that must be set, if in following condition. Set this address in uncached area. ● Drawing Raw format image

	● Using work buffer B
size_t work_buffer_size	Size (bytes) of work buffer used internally Set this with "work_buffer_address" See section 6.9.14.3, R_RGA_CalcWorkBufferSize.
int_fast32_t max_height_of_frame_buffer	Maximum height of frame buffer to be drawn Set this with "work_buffer_address"
graphics_quality_flags_t quality_flags	Drawing quality; see section 6.6.5.4.
r8g8b8a8_t background_color	Background color; see section 6.9.1.25.
bool_t is_fast_manual_flush	Whether the mode is fast manual flush mode (section 6.11.4)
void* work_buffer_b_address	Work buffer B starting address that must be set, if in following condition. ● X coord value of left of JPEG image is not alignment by 4, when frame buffer was 16 bit color. ● X coord value of left of JPEG image is not alignment by 2, when frame buffer was 32 bit color. ● Drawing JPEG image with matrix
size_t work_buffer_b_size	Size (bytes) of work buffer B Set this with "work_buffer_b_address" See section 6.9.14.4, R_RGA_CalcWorkBufferB_Size.
graphics_jpeg_decoder_t jpeg_decoder	Using JPEG decoder
size_t work_size_for_libPNG	Size (bytes) of memory area as heap area allocated by PNG decoding. This memory area is allocated in work buffer B.
bit_flags32_t internal_event_value	The value of thread attached event. Default value is R_OSPL_A_FLAG(=0x0001). Application can not wait this value. See application note of OSPL.
BSP_CFG_USER_LOCKING_TYPE* lock_object	Lock object managing right to use RGA. Lock area is from R_GRAPHICS_Initialize function to R_GRAPHICS_Finalize function. See application note of OSPL.
int_fast32_t channel_of_DMAC_1	Channel number of DMAC used in RGA.
int_fast32_t channel_of_DMAC_2	Channel number of DMAC used in RGA.
int_fast32_t event_value_of_DMAC_1	Flag of notification event as the end of DMA transfer.
int_fast32_t event_value_of_DMAC_2	Flag of notification event as the end of DMA transfer.

6.6.5.4 graphics_quality_flags_t

This is a defaultable flag indicating the drawing quality. See section 6.11.8, Defaultable Flags.

```
#include "RGA.h"
```

Constant Name	Value	Description
GRAPHICS_RENDERING_QUALITY_ANTIALIASED	1	Enable the antialiasing at the borders

GRAPHICS_RENDERING_QUALITY_NONANTIALIASED	1<<16	Disable the antialiasing at the borders
GRAPHICS_IMAGE_QUALITY_ANTIALIASED	2	Enable the interpolation filter for image
GRAPHICS_IMAGE_QUALITY_NONANTIALIASED	2<<16	Disable the interpolation filter for image

6.6.5.5 window_surfaces_t

window_surfaces_t structure	
Outline	This is a sample type of the frame buffer and screen display
Header	RGA.h, window_surfaces.h
Description	For the function equivalent to the member function, see section 6.9.9.
Member Variable	Access Inhibit

6.6.5.6 window_surfaces_config_t

window_surfaces_config_t structure

Outline: This is a type that sets window_surfaces_t.

Header: RGA.h, window_surfaces.h

Description

Member Variable:

Symbol	Description
bit_flags_fast32_t flags	For flags, see section 6.11.7, Flagged Structure Parameters. F_WINDOW_SURFACES_PIXEL_FORMAT F_WINDOW_SURFACES_LAYER_COUNT F_WINDOW_SURFACES_BUFFER_HEIGHT F_WINDOW_SURFACES_BACKGROUND_COLOR
pixel_format_t pixel_format	Pixel format of showing window. See section 6.6.5.10. [Condition] The following value can be set for the sample of RZ/A1H. ● PIXEL_FORMAT_ARGB8888 ● PIXEL_FORMAT_XRGB8888 ● PIXEL_FORMAT_RGB565 (default) ● PIXEL_FORMAT_ARGB4444 ● PIXEL_FORMAT_ARGB1555 ● PIXEL_FORMAT_YCbCr422 ● PIXEL_FORMAT_CLUT1 ● PIXEL_FORMAT_CLUT4 ● PIXEL_FORMAT_CLUT8
int_fast32_t layer_count	Count of creating layer. Default is 1. If count was 2, all pixel formats of layers are same. But, If pixel format was "PIXEL_FORMAT_YCbCr422", overlay layer's format is "PIXEL_FORMAT_ARGB4444"
int_fast32_t buffer_height	Height of the frame buffer. Default is the height of the screen.
r8g8b8a8_t background_color	Background color. See 6.6.5.19 and 6.9.1.25. Default is "R_RGA_Get_R8G8B8A8(255, 255, 255, 0)".

6.6.5.7 layer_attributes_t

layer_attributes_t structure

Outline	The following lists parameter types of the "R_WINDOW_SURFACES _AccessLayerAttributes" function
Header	RGA.h, window_surfaces.h
Description	See Layer structure of sample screen controller

Member Variable

Symbol	Description
access_t access	Whether to set or acquire values (essential)
bit_flags_fast32_t flags	Section 6.11.7, Flagged Structure Parameters (essential) Above value logical or. (e.g.) F_LAYER_ID
int_fast32_t id	Layer number to be accessed (-1: background) (essential). See 6.11.6.
int_fast32_t priority	Reserved
bool_t is_show	Reserved
r8g8b8a8_t color_key	Reserved
bool_t is_color_key	Reserved
r8g8b8a8_t layer_color	Color of entire layer (available only for ID = -1). See 6.6.5.19 and 6.9.1.25.
int_fast32_t x	Left position of layer window (pixel)
int_fast32_t y	Top position of layer window (pixel)
int_fast32_t width	Width of layer window (pixel)
int_fast32_t height	Height of layer window (pixel)
int_fast32_t offset_x	X coord in frame buffer at left end of layer window
int_fast32_t offset_y	Y coord in frame buffer at top end of layer window
uint32_t* CLUT	Array of colors to set to CLUT (Color Look Up Table). (optional) Set "CLUT" of graphics_image_properties_t (section 6.6.5.14). Pixel format is ARGB8888. If CLUT was overwritten for second image, first image using CLUT is not shown correctly.
int_fast32_t CLUT_count	Count of elements of CLUT. Set this with "CLUT" variable. Set "CLUT_count" of graphics_image_properties_t (section 6.6.5.14). [Setting range] Max is 256(CLUT8), 16(CLUT4), 2(CLUT1).

6.6.5.8 access_t

This is a type that specifies Write (setting set values) or Read (acquiring set values).

Constant Name	Value	Description
ACCESS_READ	1	Read values
ACCESS_WRITE	2	Write values

6.6.5.9 byte_per_pixel_t

```
#include "RGA.h"
typedef int byte_per_pixel_t;
```

This is a type of number of bytes per pixel. See Table 1-2.

When one pixel is less than one byte (BitPerPixel < 8), a value shifted from the number of bits per pixel is set.

When 1 pixel is 1 byte or more:

15	8	7	0
0	0	Number of bytes	

When 1 pixel is less than 1 byte:

15	8	7	0
0	Number of bits	0	

For related functions, see section 6.9.10.

6.6.5.10 pixel_format_t

This is a pixel format type.

Refer to Table 1-3 Combinations of Available Pixel Formats of Images and Pixel Formats of Drawing Destination

```
#include "RGA.h" /* or "frame_buffer.h" */
```

Constant Name	Value	Value	Description
PIXEL_FORMAT_UNKNOWN	0	0x00	Unknown
PIXEL_FORMAT_ARGB8888	1	0x01	ARGB8888
PIXEL_FORMAT_RGB565	3	0x03	RGB565
PIXEL_FORMAT_ARGB4444	5	0x05	ARGB4444
PIXEL_FORMAT_A1	13	0x0D	1bit alpha
PIXEL_FORMAT_A4	14	0x0E	4bit alpha
PIXEL_FORMAT_A8	11	0x0B	8bit alpha
PIXEL_FORMAT_RGB888	15	0x0F	RGB888 (Reserved)
PIXEL_FORMAT_R8G8B8A8	6 | (1 << 4)	0x16	R8G8B8A8
PIXEL_FORMAT_XRGB8888	0 | (1 << 6)	0x40	XRGB8888
PIXEL_FORMAT_ARGB1555	4 | (1 << 6)	0x44	ARGB1555
PIXEL_FORMAT_YCbCr422	2 | (1 << 16)	0x10002	YCbCr422
PIXEL_FORMAT_YUV422	2 | (1 << 16)	0x10002	YUV422
PIXEL_FORMAT_YUV422_GRAY_SCALE_IS_0x80	2 | (1 << 16)	0x10002	YCbCr422 Cb, Cr = 0x80 is gray
PIXEL_FORMAT_JPEG	12 | (2 << 8)	0x20C	JPEG
PIXEL_FORMAT_PNG	12 | (3 << 8)	0x30C	PNG
PIXEL_FORMAT_GIF	12 | (4 << 8)	0x40C	GIF (Reserved)
PIXEL_FORMAT_CLUT1	12 | (1 << 12)	0x100C	1 bit CLUT
PIXEL_FORMAT_CLUT4	12 | (4 << 12)	0x400C	4 bit CLUT
PIXEL_FORMAT_CLUT8	12 | (8 << 12)	0x800C	8 bit CLUT

6.6.5.11 frame_buffer_delegate_t

This is a class of objects referenced from the delegate member variable of frame_buffer_t.

This class can be defined by libraries that use frame_buffer_t or by the application.

This class is defined as follows by default. Therefore, frame_buffer_t::delegate is a void* type.

```
#include "RGA.h"
typedef void frame_buffer_delegate_t;
```

To change the type, define the type before performing #include for the header file that defines frame_buffer_delegate_t, and then perform #define frame_buffer_delegate_t frame_buffer_delegate_t.

```
typedef  MyFrameBufferClass       frame_buffer_delegate_t;
#define  frame_buffer_delegate_t  frame_buffer_delegate_t

#include  "RGA.h"  /* default define frame_buffer_delegate_t */
```

6.6.5.12 v_sync_t

v_sync_t structure	
Outline	Synchronization with the V-Sync signal of the display screen.
Header	RGA.h, vsync.h
Description	Function equivalent to the member function: See section 6.9.11.
Member Variable	Access Inhibit

6.6.5.13 vram_ex_stack_t

vram_ex_stack_t structure	
Outline	This is a sample code of stack class for off-screen buffer in external RAM
Header	RGA.h, RGA_SampleLib_Port.h
Description	Function equivalent to the member function: See section 6.9.12. For RZ/A1H, Video Display Controller 5 (VDC5) must not show the frame buffer in external RAM. Show the frame buffer in internal RAM copied from external RAM.
Member Variable	Access Inhibit

6.6.5.14 graphics_image_properties_t

graphics_image_properties_t structure

Outline: This is a type of image properties.

Header: RGA.h

Description: Related function : (6.9.2.6) R_GRAPHICS_IMAGE_GetProperties
If the source image is created in the program, call "R_GRAPHICS_IMAGE_InitByShareFrameBuffer" function (6.9.2.5).

Member Variable:

Symbol	Description
int_fast32_t width	Width of image (pixel)
int_fast32_t height	Height of image (pixel)
uint8_t* data	First address of array of image pixels. If the image format is not R8G8B8A8, this value is NULL
void* pixels	First address of array of image pixels. Even if the image format is not R8G8B8A8, this value is available.
pixel_format_t pixelFormat	Pixel format
uint32_t* CLUT	CLUT (Color Look Up Table, Palette). Array of colors. If the image does not have CLUT, this value is NULL. Set to "layer_attributes_t" (6.6.5.7).
int_fast32_t CLUT_count	Count of array elements of CLUT. Set to "layer_attributes_t" (6.6.5.7).

6.6.5.15 graphics_composite_operation_t

This is a calculation method type for alpha blend.

```
#include  "RGA.h"
```

Constant Name	Value	Description
GRAPHICS_SOURCE_OVER	1	Performs calculation (SRC over DST) of general alpha blend.

		Whether the calculation formula is the premultiplied alpha expression depends on whether alpha is present in the flag included in the header of an image to be drawn or in the drawing target pixel format. (default)
GRAPHICS_DESTINATION_OUT	7	This type is used for drawing the inverted alpha mask. Only the alpha component of drawing destination varies according to the following expression. a_dst = a_dst * (1 - a_src) When drawing an image on a frame buffer without alpha component, GRAPHICS_DESTINATION_OUT cannot be set. When the final drawing destination is XRGB8888, draw the inverted alpha mask on the back buffer of ARGB8888. Set 255 for the R_GRAPHICS_SetGlobalAlpha function or the globalAlpha property.
GRAPHICS_COPY	0	Performs no alpha blend and copy from source image data. It may be fast at that rate. For faster, when PNG image with alpha channel was decoded to the back buffer, if CRAPHICS_COPY was specified, it is not necessary to clear the back buffer on ahead.

6.6.5.16 graphics_status_t

graphics_status_t structure

Outline	This is an area type that stores graphics drawing context.
Header	RGA.h
Description	Function equivalent to the member function: ● 6.9.1.11 R_GRAPHICS_Save ● 6.9.1.12 R_GRAPHICS_Restore
Member Variable	Access Inhibit

6.6.5.17 graphics_matrix_float_t

This is a matrix element type.

```
#include "RGA.h"
typedef float graphics_matrix_float_t;
```

6.6.5.18 repetition_t

This is a type that specifies the pattern repetition method.

```
#include "RGA.h"
```

Constant Name	Value	Description
GRAPHICS_REPEAT	1	Repeating

6.6.5.19 r8g8b8a8_t

r8g8b8a8_t structure

Outline	Pixel format in which bytes are arranged in the order of red, green, blue, and alpha		
Header	RGA.h		
Description	[Setting range] A value of 0 (black, transparent) to 255 (light, opaque) is specified for each of red, green, blue, and alpha.		
Member Variable	Symbol		Description
	uint8_t	u.red	Red
	uint8_t	u.green	Green
	uint8_t	u.blue	Blue

uint8_t u.alpha	Alpha, opaque level

6.6.5.20 animation_timing_function_t

animation_timing_function_t structure	
Outline	This is an object describing timing of animation (Bezier function).
Header	RGA.h
Description	For the function equivalent to the member function, see section 6.9.13.
Member Variable	Access Inhibit

6.6.5.21 graphics_jpeg_decoder_t

Constant value of a kind of JPEG decoder.

```
#include "RGA.h"
```

Constant Name	Value	Description
GRAPHICS_JPEG_DECODER_NONE	0	Not used JPEG decoder. JPEG hardware decoder can be used from any other than RGA.
GRAPHICS_JPEG_DECODER_HARD	1	JPEG hardware decoder. (default)

6.6.5.22 graphics_async_status_t

graphics_async_status_t structure

Outline	Status of drawing graphics and interrupt status
Header	RGA.h
Description	

Member Variable

Symbol	Description
r_ospl_async_state_t State	Status of drawing graphics. Inhibit write access. This value will update by calling R_GRAPHICS_GetAsyncStatus function. See r_ospl_async_state_t type - OS Porting Layer "OSPL" Sample Program
Others are access inhibit	

6.6.6 String Format

6.6.6.1 #rrggbb, #rgb

This format describes a color by hex number of CSS Color format

Target : fillStyle (6.8.1.3)

e.g.) "#FFFF00" : Red component is 255, Green component is 255, Blue component is 0 and Alpha component is 1.0

e.g.) "#FF0" : Red component is 255, Green component is 255, Blue component is 0 and Alpha component is 1.0

6.6.7 Type of porting layers

6.6.7.1 List

Type	Description
NCGvoid	Abstract of void type for RGPNCG
NCGenum	Abstract of enumeration type for RGPNCG [6]
NCGboolean	Abstract of _Bool type for RGPNCG
NCGbitfield	Abstract of bit flags type for RGPNCG [6]
NCGchar	Abstract of string type for RGPNCG
NCGint8	Abstract of int8_t type for RGPNCG [6]
NCGint16	Abstract of int16_t type for RGPNCG [6]
NCGint32	Abstract of int32_t type for RGPNCG
NCGint64	Abstract of int64_t type for RGPNCG [6]
NCGuint8	Abstract of uint8_t type for RGPNCG [6]
NCGuint16	Abstract of uint16_t type for RGPNCG [6]
NCGuint32	Abstract of uint32_t type for RGPNCG
NCGuint64	Abstract of uint64_t type for RGPNCG [6]
NCGfloat32	Abstract of IEEE 754 single precision floating point number type for RGPNCG [6]
NCGfloat64	Abstract of IEEE 754 double precision floating point number type for RGPNCG [6]
NCGsizei	Unsigned integer type described a size for RGPNCG [6]
NCGclampf	Abstract of floating point number type let the value from 0.0 to 1.0 for RGPNCG [6]
NCGfixed	Abstract of fixed point number type for RGPNCG [6]

[6] Not used in RGA

6.6.8 Changing state of class

Figure 6-1 shows a Figure of changing state of graphics_t class, and Table 6-1 shows List of functions of each classes having R_*_Finalize member function. The function not changing state can be called at Normal state.

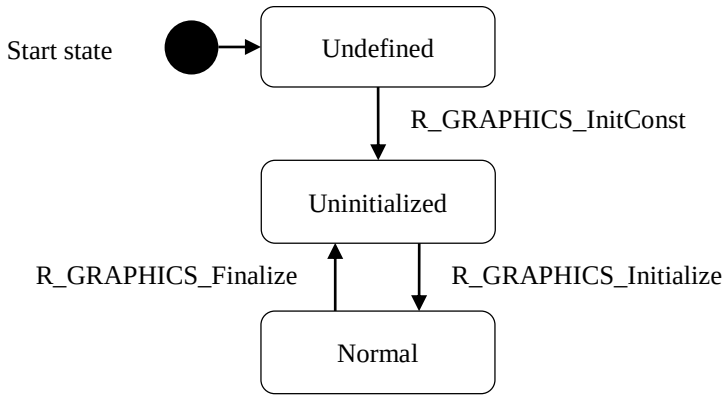


Figure 6-1 Figure of changing state of graphics_t class

Table 6-1 List of functions of each classes having R_*_Finalize member function

Class	Undefined to Uninitialized	Uninitialized to Normal	Normal to Uninitialized
graphics_t	R_GRAPHICS_InitConst	R_GRAPHICS_Initialize	R_GRAPHICS_Finalize

Figure 6-2 shows a Figure of changing state of graphics_pattern_t class, and Table 6-2 shows List of functions of each classes not having R_*_Finalize member function. The function not changing state can be called at Normal state.

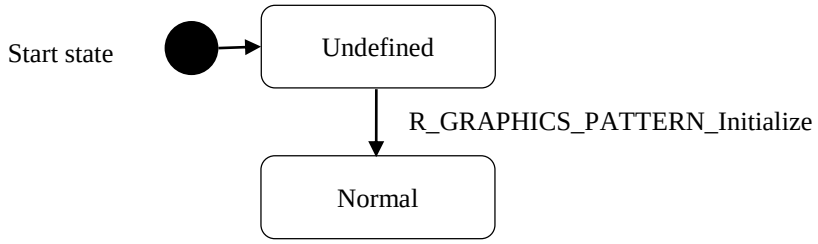


Figure 6-2 Figure of changing state of graphics_pattern_t class

Table 6-2 List of functions of each classes not having R_*_Finalize member function

Class	Undefined to Normal
graphics_image _t	R_GRAPHICS_IMAGE_InitR8G8B8A8 R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8 R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8 R_GRAPHICS_IMAGE_InitByShareFrameBuffer The object created by ImagePackager is Normal state at the start state.
graphics_pattern_t	R_GRAPHICS_PATTERN_Initialize

6.7 List of Variables

Table 6-3 shows Global Variables, Table 6-4 shows Static-Type Variables, and Table 6-5 shows Const-Type Variables.

Table 6-3 Global Variables

Type	Variable	Description	Applicable Function
None			

Table 6-4 Static-Type Variables

Type	Variable	Description	Applicable Function
None			

Table 6-5 Const-Type Variables

Type	Variable	Description	Applicable Function
None			

6.8 Properties

6.8.1 Canvas2D_ContextClass Properties

6.8.1.1 List of Properties

Section	Property	Description
6.8.1.2	c_LanguageContext	Context available for the C language API
6.8.1.3	fillStyle	Fill style
6.8.1.4	globalAlpha	One alpha value (opacity) multiplied by all drawings
6.8.1.5	globalCompositeOperation	Calculation method for alpha blend

6.8.1.2 c_LanguageContext

```
graphics_t*  Canvas2D_ContextClass::c_LanguageContext;  /* get only */
```

This property is a context available for the C language API.

Refer to this property when using a function that is provided by the C language API but is not provided by the C++API.

6.8.1.3 fillStyle

```
char* Canvas2D_ContextClass::fillStyle;  /* set only */ /* CSS color */
Canvas2D_PatternClass  Canvas2D_ContextClass::fillStyle;  /* set only */
```

This property specifies a fill method. This property has the value of either type shown below.

For the char* type, this property specifies the single-color fill color expressed by CSS Color.

For the Canvas2D_PatternClass type, this property specifies pattern.

When the char* type is specified, the fill method is single-color fill. Refer to #rrggbb, #rgb (6.6.6.1), rgb (6.9.15.2), rgba (6.9.15.3)

The initial value is opaque black.

Use the fillRect method for drawing patterns.

6.8.1.4 globalAlpha

```
float32_t  Canvas2D_ContextClass::globalAlpha;  /* get,set */
```

This property retains a single alpha value (opacity) to be multiplied by all drawings.

The default value is 1.0.

If a value less than 0.0 is set, 0.0 is retained.

If a value larger than 1.0 is set, 1.0 is retained.

This property affects the following drawing functions.

- Figure fill functions such as fillRect (Canvas2D_ContextClass)
- Pattern drawing functions such as fillRect (Canvas2D_ContextClass)
- Border drawing functions such as strokeRect (Canvas2D_ContextClass)
- Image drawing functions such as drawImage (Canvas2D_ContextClass)

This property does not affect the following drawing function.

- clearRect (Canvas2D_ContextClass)

6.8.1.5 globalCompositeOperation

```
char*  Canvas2D_ContextClass::globalCompositeOperation;  /* get,set */
```

This property retains the calculation method for alpha blend. See section 6.6.5.15.

6.8.2 Canvas2D_ImageClass Properties

6.8.2.1 List of Properties

Section	Property	Description
6.8.2.2	src	Image data structure
6.8.2.3	width	Image width
6.8.2.4	height	Image height
6.8.2.5	data	Array containing pixel color components

6.8.2.2 src

```
graphics_image_t  Canvas2D_ImageClass::src;  /* set only */
```

This property retains the raw image data structure created by ImagePackager.

6.8.2.3 width

```
int_t  Canvas2D_ImageClass::width;  /* get only */
```

This property retains the image width.

6.8.2.4 height

```
int_t  Canvas2D_ImageClass::height;  /* get only */
```

This property retains the image height.

6.8.2.5 data

```
uint8_t*  Canvas2D_ImageClass::data;  /* get only */
```

This property is an array containing pixel color components.

Upper left is the top.

Color components are arranged in the order of R, G, B, and A.

R, G, B, A: 0 to 255

6.9 Functions and Methods

6.9.1 Functions Equivalent to graphics_t Class Member Function

6.9.1.1 List of Functions

Section	Function Name	Description
6.9.1.2	R_GRAPHICS_InitConst	Initializes the constant part.
6.9.1.3	R_GRAPHICS_Initialize	Initializes the graphics drawing context object.
6.9.1.4	R_GRAPHICS_Finalize	Finalizes the graphics drawing context object.
6.9.1.5	R_GRAPHICS_SetFrameBuffer	Changes the drawing target.
6.9.1.6	R_GRAPHICS_GetFrameBuffer	Gets the drawing target.
6.9.1.7	R_GRAPHICS_Finish	Waits until drawing is completed.
6.9.1.8	R_GRAPHICS_FinishStart	Starts to wait for drawing
6.9.1.9	R_GRAPHICS_GetAsyncStatus	Gets the status of asynchronous operation and interrupt
6.9.1.10	R_GRAPHICS_OnInterrupting	Receives interrupts of drawing hardware
6.9.1.11	R_GRAPHICS_Save	Saves the set value of context to the specified status structure.
6.9.1.12	R_GRAPHICS_Restore	Returns the status structure content to context.
6.9.1.13	R_GRAPHICS_ResetMatrix	Resets the target matrix of the matrix calculation function to the unit matrix.
6.9.1.14	R_GRAPHICS_SetMatrix_2x3	Sets each element of the matrix. (2x3)
6.9.1.15	R_GRAPHICS_SetMatrix_3x3	Sets each element of the matrix. (3x3)
6.9.1.16	R_GRAPHICS_GetMatrix_3x3	Acquires each element of the matrix.
6.9.1.17	R_GRAPHICS_TranslateMatrixI	Translates matrix from the current matrix. (integer type specified)
6.9.1.18	R_GRAPHICS_TranslateMatrix	Translate matrix from the current matrix. (floating-point type specified)
6.9.1.19	R_GRAPHICS_ScaleMatrix	Enlarges or reduces matrix from the current matrix.
6.9.1.20	R_GRAPHICS_RotateMatrixDegree	Rotates matrix from the current matrix. Rotation center: (0,0)
6.9.1.21	R_GRAPHICS_ShearMatrix	Makes shear deformation from the current matrix.
6.9.1.22	R_GRAPHICS_TransformMatrix	Multiplies the current matrix by the specified 2x3 matrix.
6.9.1.23	R_GRAPHICS_MultiplyMatrix	Multiplies the current matrix by the specified 3x3 matrix.
6.9.1.24	R_GRAPHICS_GetProjectiveMatrix	Acquires a matrix that deforms a random profile quadrangle to a random profile quadrangle.
6.9.1.25	R_GRAPHICS_SetBackgroundColor	Sets the background color.
6.9.1.26	R_GRAPHICS_GetBackgroundColor	Acquires the background color.
6.9.1.27	R_GRAPHICS_GetClearColor	Acquires the color used for R_GRAPHICS_Clear.
6.9.1.28	R_GRAPHICS_Clear	Clears rectangle area.
6.9.1.29	R_GRAPHICS_DrawImage	Draws an image.
6.9.1.30	R_GRAPHICS_DrawImageResized	Enlarges or reduces an image and draws it in a rectangle.
6.9.1.31	R_GRAPHICS_DrawImageChild	Draws a part of an image.
6.9.1.32	R_GRAPHICS_FillRect	Fills a rectangle.
6.9.1.33	R_GRAPHICS_SetFillColor	Sets the color used for single-color fill for the current fill paint object.
6.9.1.34	R_GRAPHICS_SetFillPattern	Sets a pattern for the current fill paint object.
6.9.1.35	R_GRAPHICS_BeginPath	Resets the default path content to null.
6.9.1.36	R_GRAPHICS_Rect	Adds a rectangle to the default path.
6.9.1.37	R_GRAPHICS_Clip	Sets the shape of the current default path to a clipping area.
6.9.1.38	R_GRAPHICS_SetGlobalAlpha	Sets an alpha value (opacity) to be multiplied by all drawings.

6.9.1.39	R_GRAPHICS_GetGlobalAlpha	Acquires an alpha value (opacity) to be multiplied by all drawings.
6.9.1.40	R_GRAPHICS_SetGlobalCompositeOperation	Sets the calculation method for alpha blend.
6.9.1.41	R_GRAPHICS_GetGlobalCompositeOperation	Acquires the calculation method for alpha blend.
6.9.1.42	R_GRAPHICS_STATIC_SetOnInitialize	Registers a callback function that sets the default graphics_config_t value.
6.9.1.43	R_GRAPHICS_STATIC_SetOnFinalize	Registers a function that releases the memory allocated in the R_GRAPHICS_OnInitialize_FuncType function.
6.9.1.44	R_GRAPHICS_SetQualityFlags	Sets the drawing quality.
6.9.1.45	R_GRAPHICS_GetQualityFlags	Acquires the current drawing quality.
6.9.1.46	R_GRAPHICS_SetStrokeColor	Sets the color used for single-color fill for the paint object of the current border.
6.9.1.47	R_GRAPHICS_StrokeRect	Draws sides of rectangle.
6.9.1.48	R_GRAPHICS_BeginSoftwareRendering	Notifies the graphics library of the start of software rendering.
6.9.1.49	R_GRAPHICS_EndSoftwareRendering	Notifies the graphics library of the end of software rendering.
6.9.1.50	R_GRAPHICS_EndRenderingInFin	Calls this function from the end of the function that performs software rendering.
6.9.1.51	R_GRAPHICS_BeginSoftwareRendering2	R_GRAPHICS_BeginSoftwareRendering for the memory on L2 cache.
6.9.1.52	R_GRAPHICS_BeginSoftwareRenderingA	Calls R_GRAPHICS_BeginSoftwareRendering2 depending on the address.

6.9.1.2 R_GRAPHICS_InitConst

R_GRAPHICS_InitConst		
Outline	Initializes the constant part used by the RGA.	
Header	RGA.h	
Declaration	void R_GRAPHICS_InitConst(graphics_t* self);	
Description	Refer to section 6.11.9, Function to Initialize Internal Variables with Constants (*_initConst Function).	
Argument	graphics_t* self	Graphics drawing context
Return value	None	

6.9.1.3 R_GRAPHICS_Initialize

R_GRAPHICS_Initialize		
Outline	Initializes the graphics drawing context object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Initialize (graphics_t* self, graphics_config_t* config);	
Description	Initializes internal variables. Initializes Renesas Graphics Processor for OpenVG™ (R-GPVG2) and JPEG Codec Unit (JCU). For the built-in version, register the interrupt handler before calling this function. When "self" is no longer be used, call R_GRAPHICS_Finalize. There is only one context. When two or more frame buffers was drawn, change the frame buffer by "R_GRAPHICS_SetFrameBuffer" function.	
Argument	graphics_t* self	Graphics drawing context
	graphics_config_t* config	See section 6.6.5.2.
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.4 R_GRAPHICS_Finalize

R_GRAPHICS_Finalize		
Outline	Finalizes the graphics drawing context object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Finalize(graphics_t* self, errnum_t e);	
Description		
Argument	graphics_t* self	Graphics drawing context
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code or e 0 = successful and e = 0	

6.9.1.5 R_GRAPHICS_SetFrameBuffer

R_GRAPHICS_SetFrameBuffer		
Outline	Changes the drawing target.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetFrameBuffer(graphics_t* self, frame_buffer_t* frame_buffer);	
Description		
Argument	graphics_t* self	Graphics drawing context
	frame_buffer_t* frame_buffer	Frame buffer to be drawn
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.6 R_GRAPHICS_GetFrameBuffer

R_GRAPHICS_GetFrameBuffer

Outline	Gets the drawing target.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_GetFrameBuffer(graphics_t* self, frame_buffer_t** out_frame_buffer);	
Description		
Argument	graphics_t* self	Graphics drawing context
	frame_buffer_t** out_frame_buffer	(Output) Frame buffer to be drawn
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.7 R_GRAPHICS_Finish

R_GRAPHICS_Finish		
Outline	Waits until drawing is completed.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Finish(graphics_t* self);	
Description	This waits for finish drawing by RGA or by the application directly to the frame buffer and changes to the state to able to show the frame buffer. In fast manual flush mode (see section 6.11.4), when the CPU directly reads or writes frame buffer data in the cache area or non-cache area after the RGA API is called, enclose by calling R_GRAPHICS_BeginSoftwareRendering to R_GRAPHICS_EndSoftwareRendering. These functions wait hardware rendering and flush the CPU cache only if necessary. If the process waits for finish drawing asynchronously, call "R_GRAPHICS_FinishStart" function.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.8 R_GRAPHICS_FinishStart

R_GRAPHICS_FinishStart		
Outline	Starts to wait for drawing	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_FinishStart(graphics_t* self, r_ospl_async_t* Async);	
Description	This is asynchronous version of R_GRAPHICS_Finish function. See R_DRIVER_TransferStart function - OS Porting Layer "OSPL" Sample Program.	
Argument	graphics_t* self	Graphics drawing context
	r_ospl_async_t* Async	(Input/Output) Notify setting. NULL is not permitted
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.9 R_GRAPHICS_GetAsyncStatus

R_GRAPHICS_GetAsyncStatus		
Outline	Gets the status of asynchronous operation and interrupt	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_GetAsyncStatus(graphics_t* self, const graphics_async_status_t** out_Status);	
Description	See R_DRIVER_GetAsyncStatus function - OS Porting Layer "OSPL" Sample Program.	
Argument	graphics_t* self	Graphics drawing context
	const graphics_async_status_t** out_Status	(Output) Pointer to the structure of interrupt and asynchronous operation
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.10 R_GRAPHICS_OnInterrupting

R_GRAPHICS_OnInterrupting	
Outline	Receives interrupts of drawing hardware
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_OnInterrupting(void);
Description	See R_DRIVER_OnInterrupting function - OS Porting Layer "OSPL" Sample Program.
Argument	None
Return value	Error code. If there is no error, the return value is 0.

6.9.1.11 R_GRAPHICS_Save

R_GRAPHICS_Save		
Outline	Saves the set value of context to the specified status structure.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Save(graphics_t* self, graphics_status_t* out_status);	
Description		
Argument	graphics_t* self	Graphics drawing context
	graphics_status_t* out_status	(Output) Set value of context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.12 R_GRAPHICS_Restore

R_GRAPHICS_Restore		
Outline	Returns the status structure content to context.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Restore(graphics_t* self, graphics_status_t* status, errnum_t e);	
Description		
Argument	graphics_t* self	Graphics drawing context
	graphics_status_t* status	Set value of context
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code or e 0 = successful and e = 0	

6.9.1.13 R_GRAPHICS_ResetMatrix

R_GRAPHICS_ResetMatrix		
Outline	Resets the matrix to be a matrix calculation function target to the unit matrix.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_ResetMatrix(graphics_t* self);	
Description		
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.14 R_GRAPHICS_SetMatrix_2x3

R_GRAPHICS_SetMatrix_2x3	
Outline	Sets each element of the current matrix. (2x3)
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_SetMatrix_2x3(graphics_t* self, graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty);
Description	When the library was ported, take care computing error.
Argument	graphics_t* self Graphics drawing context graphics_matrix_float_t sx, 2x3 matrix graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty $\begin{pmatrix} sx & kx & tx \\ ky & sy & ty \\ 0 & 0 & 1 \end{pmatrix}$
Return value	Error code. If there is no error, the return value is 0.

6.9.1.15 R_GRAPHICS_SetMatrix_3x3

R_GRAPHICS_SetMatrix_3x3	
Outline	Sets each element of the current matrix (Matrix[]) (3x3).
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_SetMatrix_3x3(graphics_t* self, graphics_matrix_float_t * matrix);
Description	When the library was ported, take care computing error.
Argument	graphics_t* self Graphics drawing context graphics_matrix_float_t * matrix 3x3 matrix (array) $\begin{pmatrix} Matrix[0] & Matrix[3] & Matrix[6] \\ Matrix[1] & Matrix[4] & Matrix[7] \\ Matrix[2] & Matrix[5] & Matrix[8] \end{pmatrix}$
Return value	Error code. If there is no error, the return value is 0.

6.9.1.16 R_GRAPHICS_GetMatrix_3x3

R_GRAPHICS_GetMatrix_3x3	
Outline	Acquires each element of the current matrix (Matrix[]).
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_GetMatrix_3x3(graphics_t* self, graphics_matrix_float_t * out_matrix);
Description	
Argument	graphics_t* self Graphics drawing context graphics_matrix_float_t * out_matrix (Output) 3x3 matrix (array) $\begin{pmatrix} Matrix[0] & Matrix[3] & Matrix[6] \\ Matrix[1] & Matrix[4] & Matrix[7] \\ Matrix[2] & Matrix[5] & Matrix[8] \end{pmatrix}$
Return value	Error code. If there is no error, the return value is 0.

6.9.1.17 R_GRAPHICS_TranslateMatrixI

R_GRAPHICS_TranslateMatrixI		
Outline	Translates the current matrix (M). (Integer type specified)	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_TranslateMatrixI(graphics_t* self, int_t tx, int_t ty);	
Description	$M = M \cdot \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$	
Argument	graphics_t* self	Graphics drawing context
	int_t tx, int_t ty	Displacement (When the origin is at the upper left, plus of X means right direction and plus of Y means downward direction.)
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.18 R_GRAPHICS_TranslateMatrix

R_GRAPHICS_TranslateMatrix		
Outline	Translates the current matrix (M). (Floating-point type specified)	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_TranslateMatrix(graphics_t* self, graphics_matrix_float_t tx, graphics_matrix_float_t ty);	
Description	$M = M \cdot \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t tx	Displacement (When the origin is at the upper left, plus of X means right direction and plus of Y means downward direction.)
	graphics_matrix_float_t ty	
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.19 R_GRAPHICS_ScaleMatrix

R_GRAPHICS_ScaleMatrix		
Outline	Enlarges or reduces the current matrix (M).	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_ScaleMatrix(graphics_t* self, graphics_matrix_float_t sx, graphics_matrix_float_t sy);	
Description	$M = M \cdot \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$	
	When the library was ported, take care computing error.	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t tx	Magnification (Enlargement/reduction center: Origin)
	graphics_matrix_float_t ty	
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.20 R_GRAPHICS_RotateMatrixDegree

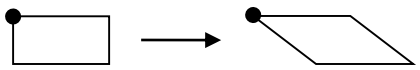
R_GRAPHICS_RotateMatrixDegree		
Outline	Rotates the current matrix (M). The center coordinates of rotation are (0,0).	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_RotateMatrixDegree(graphics_t* self, graphics_matrix_float_t angle);	
Description	$M = M \cdot \begin{pmatrix} \cos(angle) & -\sin(angle) & 0 \\ \sin(angle) & \cos(angle) & 0 \\ 0 & 0 & 1 \end{pmatrix}$	
	When the library was ported, take care computing error.	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t angle	Rotation angle (degrees) (When the origin is at the upper left, plus means clockwise direction.)
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.21 R_GRAPHICS_ShearMatrix

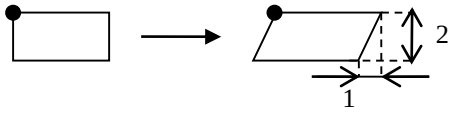
R_GRAPHICS_ShearMatrix	
Outline	Makes shear deformation of the current matrix (M).
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_ShearMatrix(graphics_t* self, graphics_matrix_float_t shx, graphics_matrix_float_t shy);
Description	$M = M \cdot \begin{pmatrix} 1 & shx & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

When (shx, shy) = (1.0, 0.0), a parallelogram is generated with perpendicular sides tilted 45 degrees.

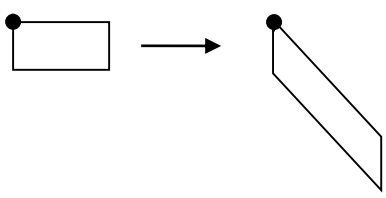
Note, however, that the matrix is shifted unless the origin is at the upper left of the rectangle.



When (shx, shy) = (-0.5, 0.0), a parallelogram is generated with hypotenuses of a triangle (base : height = 1 : 2).



When (shx, shy) = (0.0, 1.0), a parallelogram is generated with horizontal sides tilted 45 degrees.



	When the library was ported, take care computing error.	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t shx	Rate of shear (Shear center: Origin)
	graphics_matrix_float_t shy	
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.22 R_GRAPHICS_TransformMatrix

R_GRAPHICS_TransformMatrix	
Outline	Multiplies the current matrix (M) by the specified 2x3 matrix.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_TransformMatrix(graphics_t* self, graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty);
Description	$M = M \cdot \begin{pmatrix} sx & kx & tx \\ ky & sy & ty \\ 0 & 0 & 1 \end{pmatrix}$
	When the library was ported, take care computing error.
Argument	graphics_t* self Graphics drawing context graphics_matrix_float_t sx 2x3 matrix to be multiplied graphics_matrix_float_t ky graphics_matrix_float_t kx graphics_matrix_float_t sy graphics_matrix_float_t tx graphics_matrix_float_t ty
Return value	Error code. If there is no error, the return value is 0.

6.9.1.23 R_GRAPHICS_MultiplyMatrix

R_GRAPHICS_MultiplyMatrix	
Outline	Multiplies the current matrix (M) by the specified 3x3 matrix (Matrix[]).
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_MultiplyMatrix(graphics_t* self, graphics_matrix_float_t * matrix);
Description	$M = M \cdot \begin{pmatrix} Matrix[0] & Matrix[3] & Matrix[6] \\ Matrix[1] & Matrix[4] & Matrix[7] \\ Matrix[2] & Matrix[5] & Matrix[8] \end{pmatrix}$
	When the library was ported, take care computing error.
Argument	graphics_t* self Graphics drawing context graphics_matrix_float_t * matrix 2x3 matrix to be multiplied (array with nine elements)
Return value	Error code. If there is no error, the return value is 0.

6.9.1.24 R_GRAPHICS_GetProjectiveMatrix

R_GRAPHICS_GetProjectiveMatrix	
Outline	Acquires a matrix that deforms a random profile quadrangle to a random profile quadrangle.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_GetProjectiveMatrix(

	graphics_matrix_float_t source_top_left_x, graphics_matrix_float_t source_top_left_y, graphics_matrix_float_t source_top_right_x, graphics_matrix_float_t source_top_right_y, graphics_matrix_float_t source_bottom_left_x, graphics_matrix_float_t source_bottom_left_y , graphics_matrix_float_t source_bottom_right_x, graphics_matrix_float_t source_bottom_right_y, graphics_matrix_float_t destination_top_left_x, graphics_matrix_float_t destination_top_lLeft_y, graphics_matrix_float_t destination_top_right_x, graphics_matrix_float_t destination_top_right_y, graphics_matrix_float_t destination_bottom_left_x, graphics_matrix_float_t destination_bottom_left_y, graphics_matrix_float_t destination_bottom_right_x, graphics_matrix_float_t destination_bottom_right_y, graphics_matrix_float_t * out_matrix);
Description	When the library was ported, take care computing error.
Argument	graphics_matrix_float_t source* — Four-point coordinates before conversion graphics_matrix_float_t destination* — Four-point coordinates after conversion graphics_matrix_float_t * out_matrix — (Output) 3x3 matrix (array with nine elements)
Return value	Error code. If there is no error, the return value is 0.

6.9.1.25 R_GRAPHICS_SetBackgroundColor

R_GRAPHICS_SetBackgroundColor	
Outline	Sets the background color.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_SetBackgroundColor(graphics_t* self, r8g8b8a8_t color);
Description	When a frame buffer without alpha is to be drawn, the background color is the same as the clear color. The default background color is white. When a frame buffer with alpha is to be drawn, the background color is specified color and the clear color drawing to the target is transparent black even if any color specified. Set the background color for the back layer for correct look. .
Argument	graphics_t* self — Graphics drawing context r8g8b8a8_t color — Use R_RGA_Get_R8G8B8A8() to acquire the background color.
Return value	Error code. If there is no error, the return value is 0.

6.9.1.26 R_GRAPHICS_GetBackgroundColor

R_GRAPHICS_GetBackgroundColor	
Outline	Acquires the background color.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_GetBackgroundColor(graphics_t* self, r8g8b8a8_t* out_color);
Description	
Argument	graphics_t* self — Graphics drawing context r8g8b8a8_t* out_color — (Output) Background color
Return value	Error code. If there is no error, the return value is 0.

6.9.1.27 R_GRAPHICS_GetClearColor

R_GRAPHICS_GetClearColor		
Outline	Acquires the color used for R_GRAPHICS_Clear.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_GetClearColor(graphics_t* self, r8g8b8a8_t* out_color);	
Description	Use R_GRAPHICS_SetBackgroundColor() to set the clear color.	
Argument	graphics_t* self	Graphics drawing context
	r8g8b8a8_t* out_color	(Output) Clear color
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.28 R_GRAPHICS_Clear

R_GRAPHICS_Clear		
Outline	Clears the rectangle area.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Clear(graphics_t* self, int_t min_x, int_t min_y, int_t width, int_t height);	
Description	Use R_GRAPHICS_SetBackgroundColor() to set the clear color. When a double-buffer is used, the drawing buffer is cleared. Therefore, the cleared content is not displayed only by calling this function. Use R_WINDOW_SURFACES_SwapBuffers() to apply display. This function is affected by clipping.	
Argument	graphics_t* self	Graphics drawing context
	int_t min_x, int_t min_y, int_t width, int_t height	Rectangle area
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.29 R_GRAPHICS_DrawImage

R_GRAPHICS_DrawImage	
Outline	Draws an image at coordinates (min_x, min_y) with the same size.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_DrawImage(graphics_t* self, graphics_image_t* image, int_t min_x, int_t min_y);
Description	Image data to be specified for the image argument can be created by using section 7.1, Image Format Conversion by ImagePackager. JPEG data can be directly specified for arguments. See section 6.11.2, Identifying Image Format. This function is affected by R_GRAPHICS_SetGlobalAlpha. This function is also affected by the current matrix and clipping. When drawing an image in the YUV422 format, if the value converted from min_x or min_y by the matrix is not an even number, an error occurs. When the alpha component is included in the image and is not included in the drawing target frame buffer, RGB components to be drawn in the frame buffer are blended to values that have been multiplied by the alpha component. When the alpha component is included in the drawing target frame buffer, RGB components are blended to values that have not been multiplied by the alpha component. An example of pixel format including the alpha component: ARGB8888, ARGB4444, ARGB1555 An example of pixel format without alpha component: XRGB8888, RGB565, YUV422

When drawing characters of raster font, specify the image of A8, A4 and A1 (Alpha only) format to "image" argument. But, width of the image must be same as width of drawing target frame buffer for RZ/A1H. Also, the image of A8, A4 and A1 format can not be zoomed.

Specify a CLUT-format image as same bit count as the frame buffer for drawing in the CLUT format frame buffer. Only min_x = 0 and min_y = 0 can be specified as a drawing position. If source image's width was not byte unit, an error is raised. If CLUT color in the video controller was fit with drawing image, set CLUT of graphics_image_properties_t (6.6.5.14) to the video controller.

In fast manual flush mode (see section 6.11.4), when the image data specified for the image argument exists in the array variables prepared by the application, flushing is required. However, flushing is not required when ROM data is used.
When performing flush, directly flush the CPU cache or enclose the image data read/write processing by R_GRAPHICS_BeginSoftwareRendering to R_GRAPHICS_EndSoftwareRendering.

If the frame buffer which has source image was same as target frame buffer and there was overlap between the specified source and destination area, this function returns an error.

Argument	graphics_t* self	Graphics drawing context
	graphics_image_t* image	Image
	int_t min_x, int_t min_y	Minimum X and Y coordinates
Return value	Error code. If there is no error, the return value is 0.	

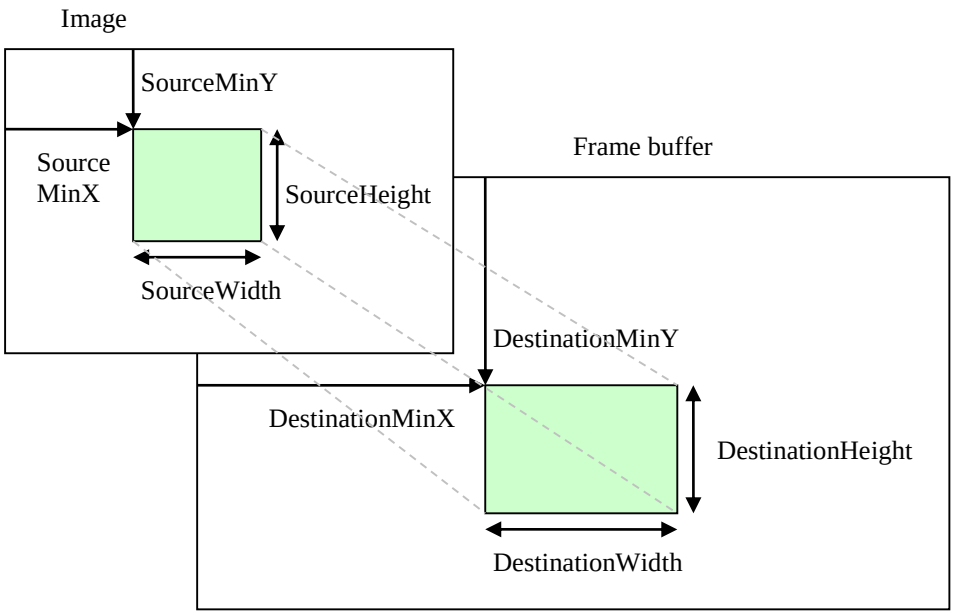
6.9.1.30 R_GRAPHICS_DrawImageResized

R_GRAPHICS_DrawImageResized		
Outline	Enlarges or reduces an image and draws it in the specified rectangle.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_DrawImageResized(graphics_t* self, graphics_image_t* image, int_t min_x, int_t min_y, int_t width, int_t height);	
Description	See the description on the R_GRAPHICS_DrawImage function. See 6.9.1.29.	
Argument	graphics_t* self	Graphics drawing context
	graphics_image_t* image	Image
	int_t min_x, int_t min_y	Minimum X and Y coordinates
	int_t width, int_t height	Width and height of the drawing target
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.31 R_GRAPHICS_DrawImageChild

R_GRAPHICS_DrawImageChild	
Outline	Draws a part of an image.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_DrawImageChild(graphics_t* self, graphics_image_t* image, int_t source_min_x, int_t source_min_y, int_t source_width, int_t source_height, int_t destination_min_x, int_t destination_min_y, int_t destination_width, int_t destination_height);

Description



When source_width ≠ destination_width or source_height ≠ destination_height, images are enlarged or reduced.

See the description on the R_GRAPHICS_DrawImage function. See 6.9.1.29.

Argument	graphics_t* self	Graphics drawing context
	graphics_image_t* image	Image
	int_t source_min_x int_t source_min_y	Minimum X and Y coordinates in the image
	int_t source_width int_t source_height	Width and height in the image
	int_t destination_min_x int_t destination_min_y	Minimum X and Y coordinates of the drawing target
	int_t destination_width int_t destination_height	Width and height of the drawing target
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.32 R_GRAPHICS_FillRect

R_GRAPHICS_FillRect		
Outline	Fills the rectangle area specified by the argument.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_FillRect(graphics_t* self, int_t min_x, int_t min_y, int_t width, int_t height);	
Description	This function is affected by the current matrix, the current fill, and clipping. No border is drawn. This function is affected by R_GRAPHICS_SetGlobalAlpha.	
Argument	graphics_t* self	Graphics drawing context
	int_t min_x, int_t min_y	Minimum X and Y coordinates of rectangle
	int_t width, int_t height	Width and height of rectangle
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.33 R_GRAPHICS_SetFillColor

R_GRAPHICS_SetFillColor	
Outline	Changes the paint object of the current fill to single-color fill and sets the fill color.
Header	RGA.h

Declaration	errnum_t R_GRAPHICS_SetFillColor(graphics_t* self, r8g8b8a8_t Color);	
Description	The initial value is opaque black.	
Argument	graphics_t* self	Graphics drawing context
	r8g8b8a8_t color	Fill color. Use R_RGA_Get_R8G8B8A8() for the fill color setting.
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.34 R_GRAPHICS_SetFillPattern

R_GRAPHICS_SetFillPattern		
Outline	Sets the pattern for the current fill paint object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetFillPattern(graphics_t* self, graphics_pattern_t* pattern);	
Description	Use R_GRAPHICS_FillRect for drawing.	
Argument	graphics_t* self	Graphics drawing context
	graphics_pattern_t* pattern	Use R_GRAPHICS_PATTERN_Initialize() to initialize the pattern object.
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.35 R_GRAPHICS_BeginPath

R_GRAPHICS_BeginPath		
Outline	Resets the default path content to null.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginPath(graphics_t* self);	
Description		
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.36 R_GRAPHICS_Rect

R_GRAPHICS_Rect		
Outline	Adds a rectangle to the default path.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Rect(graphics_t* self, int_t min_x, int_t min_y, int_t width, int_t height);	
Description	This function is used to set the clipping area.	
Argument	graphics_t* self	Graphics drawing context
	int_t min_x, int_t min_y	Minimum X and Y coordinates of rectangle
	int_t width, int_t height	Width and height of rectangle
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.37 R_GRAPHICS_Clip

R_GRAPHICS_Clip	
Outline	Sets the shape of the current default path to a clipping area.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_Clip(graphics_t* self);
Description	When the current default path is empty or is not a rectangle, an error occurs. When the current default path is empty, drawing is disabled in any area. If this function is called when the current clipping area is a part of the frame buffer, the area common to the previous clipping area and the default path becomes a new clipping area.

	To restore entire drawing, call R_GRAPHICS_Restore.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.38 R_GRAPHICS_SetGlobalAlpha

R_GRAPHICS_SetGlobalAlpha		
Outline	Sets an alpha value (opacity) to be multiplied by all drawings.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetGlobalAlpha(graphics_t* self, uint8_t alpha_value);	
Description	The default value is 255. This function affects the following drawing functions. Figure fill functions such as R_GRAPHICS_FillRect Pattern drawing function R_GRAPHICS_FillRect Border drawing functions such as R_GRAPHICS_StrokeRect Image drawing functions such as R_GRAPHICS_DrawImage This function does not affect the following drawing function. R_GRAPHICS_Clear	
Argument	graphics_t* self	Graphics drawing context
	uint8_t alpha_value	Alpha value (0 to 255). A smaller value makes drawing light.
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.39 R_GRAPHICS_GetGlobalAlpha

R_GRAPHICS_GetGlobalAlpha		
Outline	Acquires an alpha value (opacity) to be multiplied by all drawings.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_GetGlobalAlpha(graphics_t* self, uint8_t* out_alpha_value);	
Description		
Argument	graphics_t* self	Graphics drawing context
	uint8_t* out_alpha_value	(Output) Alpha value (0 to 255). A smaller value makes drawing light.
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.40 R_GRAPHICS_SetGlobalCompositeOperation

R_GRAPHICS_SetGlobalCompositeOperation		
Outline	Sets the calculation method for alpha blend.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetGlobalCompositeOperation(graphics_t* self, graphics_composite_operation_t composite_operation);	
Description	The following case other than GRAPHICS_SOURCE_OVER can be set. Image drawing such as R_GRAPHICS_DrawImage	
Argument	graphics_t* self	Graphics drawing context
	graphics_composite_operation_t composite_operation	Calculation method
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.41 R_GRAPHICS_GetGlobalCompositeOperation

R_GRAPHICS_GetGlobalCompositeOperation	
Outline	Acquires the calculation method for alpha blend.

Header	RGA.h
Declaration	errnum_t R_GRAPHICS_GetGlobalCompositeOperation(graphics_t* self, graphics_composite_operation_t* out_composite_operation);
Description	
Argument	graphics_t* self — Graphics drawing context graphics_composite_operation_t* out_composite_operation — (Output) Calculation method
Return value	Error code. If there is no error, the return value is 0.

6.9.1.42 R_GRAPHICS_STATIC_SetOnInitialize

R_GRAPHICS_STATIC_SetOnInitialize	
Outline	Registers a callback function that sets the default graphics_config_t value.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_STATIC_SetOnInitialize(R_GRAPHICS_OnInitialize_FuncType callback_function);
Description	The R_GRAPHICS_STATIC_OnInitializeDefault function is called back when this function is not called.
Argument	R_GRAPHICS_OnInitialize_FuncType callback_function — Callback function or NULL
Return value	Error code. If there is no error, the return value is 0.

6.9.1.43 R_GRAPHICS_STATIC_SetOnFinalize

R_GRAPHICS_STATIC_SetOnFinalize	
Outline	Registers a function that releases the memory allocated in the R_GRAPHICS_OnInitialize_FuncType function.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_STATIC_SetOnFinalize(R_GRAPHICS_OnFinalize_FuncType callback_function);
Description	The R_GRAPHICS_STATIC_OnInitializeDefault function is called back when this function is not called.
Argument	R_GRAPHICS_OnFinalize_FuncType callback_function — Callback function or NULL
Return value	Error code. If there is no error, the return value is 0.

6.9.1.44 R_GRAPHICS_SetQualityFlags

R_GRAPHICS_SetQualityFlags	
Outline	Sets the drawing quality.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_SetQualityFlags(graphics_t* self, graphics_quality_flags_t qualities);
Description	
Argument	graphics_t* self, — Graphics drawing context graphics_quality_flags_t qualities — See section 6.6.5.4.
Return value	Error code. If there is no error, the return value is 0.

6.9.1.45 R_GRAPHICS_GetQualityFlags

R_GRAPHICS_GetQualityFlags	
Outline	Acquires the current drawing quality.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_GetQualityFlags(graphics_t* self,

graphics_quality_flags_t* out_qualities);

Description		
Argument	graphics_t* self,	Graphics drawing context
	graphics_quality_flags_t* out_qualities	(Output) See section 6.6.5.4.
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.46 R_GRAPHICS_SetStrokeColor

R_GRAPHICS_SetStrokeColor		
Outline	Sets the color used for single-color fill for the paint object of the current border.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetStrokeColor(graphics_t* self, r8g8b8a8_t color);	
Description		
Argument	graphics_t* self,	Graphics drawing context
	r8g8b8a8_t color	Border color
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.47 R_GRAPHICS_StrokeRect

R_GRAPHICS_StrokeRect		
Outline	Draws sides of rectangle.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_StrokeRect(graphics_t* self, int_t min_x, int_t min_y, int_t width, int_t height);	
Description	This function is affected by line width, border color, and clipping. Fill is not made. When the current matrix is not the unit matrix, an error occurs. This function is affected by R_GRAPHICS_SetGlobalAlpha.	
Argument	graphics_t* self,	Graphics drawing context
	int_t min_x, int_t min_y	Minimum X and Y coordinates of rectangle
	int_t width, int_t height	Width and height of rectangle
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.48 R_GRAPHICS_BeginSoftwareRendering

R_GRAPHICS_BeginSoftwareRendering		
Outline	Notifies the graphics library of the start of software rendering.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginSoftwareRendering(graphics_t* self);	
Description	This function must be called in fast manual flush mode (see section 6.11.4). For RZ/A1H, this function is used before software rendering to the frame bufffer in On-Chip RAM. See (6.9.1.52) R_GRAPHICS_BeginSoftwareRenderingA Depending on a condition, this function waits for finishing drawing like "R_GRAPHICS_Finish" function.	
Argument	graphics_t* self,	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.49 R_GRAPHICS_EndSoftwareRendering

R_GRAPHICS_EndSoftwareRendering	
Outline	Notifies the graphics library of the end of software rendering.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_EndSoftwareRendering(graphics_t* self);
Description	This function must be called in fast manual flush mode (see section 6.11.4).

	To call this function from functions that support error processing, also call R_GRAPHICS_EndRenderingInFin at the end of the function.	
Argument	graphics_t* self,	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.50 R_GRAPHICS_EndRenderingInFin

R_GRAPHICS_EndRenderingInFin		
Outline	Call this function from the end of the function that performs software rendering.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_EndRenderingInFin(graphics_t* self, errnum_t e);	
Description	This function must be called in fast manual flush mode (see section 6.11.4).	
	To call this function from functions that support error processing, also call R_GRAPHICS_EndRenderingInFin at the end of the function.	
Argument	graphics_t* self,	Graphics drawing context
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.51 R_GRAPHICS_BeginSoftwareRendering2

R_GRAPHICS_BeginSoftwareRendering2		
Outline	R_GRAPHICS_BeginSoftwareRendering for the memory on L2 cache.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginSoftwareRendering2(graphics_t* self);	
Description	This function is usually not called.	
	This function is called in the case of RGA using RGAH, defined "R_OSPL_AXI_Get2ndCacheAttribute" function as accessing not used L2 cache from OpenVG™ hardware in RZ/A1H and fast manual flush mode (see section 6.11.4).	
	For RZ/A1H, this function is used before software rendering to the frame bufffer in SDRAM.	
	Depending on a condition, this function waits for finishing drawing like "R_GRAPHICS_Finish" function.	
Argument	graphics_t* self,	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

6.9.1.52 R_GRAPHICS_BeginSoftwareRenderingA

R_GRAPHICS_BeginSoftwareRenderingA		
Outline	Calls R_GRAPHICS_BeginSoftwareRendering2 depending on the address.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginSoftwareRenderingA(graphics_t* self, void* Address);	
Description	This function is usually not called.	
	This function is called in the case of RGA using RGAH, defined "R_OSPL_AXI_Get2ndCacheAttribute" function as accessing not used L2 cache from OpenVG™ hardware in RZ/A1H and fast manual flush mode (see section 6.11.4).	
	This function calls "R_OSPL_MEMORY_GetLevelOfFlush" function with specified address and calls "R_GRAPHICS_BeginSoftwareRendering" or "R_GRAPHICS_BeginSoftwareRendering2" depending on the level of flush.	
Argument	graphics_t* self,	Graphics drawing context
	void* Address	Buffer address to render by software
Return value	Error code. If there is no error, the return value is 0.	

6.9.2 Functions Equivalent to graphics_image_t Class Member Function

6.9.2.1 List of Functions

Section	Function Name	Description
6.9.2.2	R_GRAPHICS_IMAGE_InitR8G8B8A8	Initializes the r8g8b8a8_t image object.
6.9.2.3	R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8	Initializes the image object to the same width and height.
6.9.2.4	R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8	Initializes the image object to which a part of frame buffer being displayed is copied.
6.9.2.5	R_GRAPHICS_IMAGE_InitByShareFrameBuffer	Initializes the frame buffer data as an image.
6.9.2.6	R_GRAPHICS_IMAGE_GetProperties	Get properties of the image.

6.9.2.2 R_GRAPHICS_IMAGE_InitR8G8B8A8

R_GRAPHICS_IMAGE_InitR8G8B8A8		
Outline	Initializes the r8g8b8a8_t image object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_IMAGE_InitR8G8B8A8(graphics_image_t* self, void* image_data_array, size_t image_data_array_size, int_t width, int_t height);	
Description	Initializes internal variables. Acquirable image data is arranged in a uint8_t-type array in the order of Red, Green, Blue, and Alpha from the upper-left pixel to the lower-right pixel.	
Argument	graphics_image_t* self	Image
	void* image_data_array	Starting address of an array to be a memory area that stores images
	size_t image_data_array_size	Size (bytes) of the memory area indicated by image_data_array
	int_t width, int_t height	Width and height of image
Return value	Error code. If there is no error, the return value is 0.	

6.9.2.3 R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8

R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8		
Outline	Initializes the image object to the same width and height.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8(graphics_image_t* self, void* image_data_array, size_t image_data_array_size, graphics_image_t* same_size_image);	
Description	Initializes internal variables.	
Argument	graphics_image_t* self	Image
	void* image_data_array	Starting address of an array to be a memory area that stores images
	size_t image_data_array_size	Size (bytes) of the memory area indicated by image_data_array
	graphics_image_t* same_size_image	Image object that references width and height
Return value	Error code. If there is no error, the return value is 0.	

6.9.2.4 R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8

R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8	
Outline	Initializes the image object to which a part of frame buffer being displayed is copied.

Header	RGA.h
Declaration	errnum_t R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8(graphics_image_t* self, void* image_data_array, size_t image_data_array_size, graphics_t* context, int_t min_x, int_t min_y, int_t width, int_t height);
Description	Initializes internal variables. Acquirable image data is arranged in a uint8_t-type array in the order of Red, Green, Blue, and Alpha from the upper-left pixel to the lower-right pixel.
Argument	graphics_image _t* self — Image void* image_data_array — Starting address of an array to be a memory area that stores images size_t image_data_array_size — Size (bytes) of the memory area indicated by "image_data_array" argument graphics_t* context — Context with copy source frame buffer to be drawn int_t min_x, int_t min_y — Minimum X and Y coordinates (frame buffer coordinates) of the range to be acquired int_t width, int_t height — Width and height of the range to be acquired
Return value	Error code. If there is no error, the return value is 0.

6.9.2.5 R_GRAPHICS_IMAGE_InitByShareFrameBuffer

R_GRAPHICS_IMAGE_InitByShareFrameBuffer	
Outline	Initializes the frame buffer data as an image.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_IMAGE_InitByShareFrameBuffer(graphics_image _t* self, frame_buffer_t* frame_buffer);
Description	Initializes internal variables. The VRAM area indicated by `frame_buffer->buffer_address [frame_buffer->show_buffer_index]` at the time when this function is called is shared with image (self). [Condition] This function can not initialize CLUT format image. RGA supports static CLUT format image made from ImagePackager. [Condition] When wrong stride alignment error was raised, change width of the buffer witch becomes to fit stride alignment and copy the image to the buffer. RGA can not copy at this condition. Also, draw the source image (a part of the buffer) by calling "R_GRAPHICS_DrawImageChild" function.
Argument	graphics_image _t* self — Image frame_buffer_t* frame_buffer — Frame buffer that contains the image
Return value	Error code. If there is no error, the return value is 0.

6.9.2.6 R_GRAPHICS_IMAGE_GetProperties

R_GRAPHICS_IMAGE_GetProperties	
Outline	Get image properties.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_IMAGE_GetProperties(graphics_image _t* self, graphics_image_properties_t* out_properties);
Description	In the case of fast manual flush mode (6.11.4): Flush operation must be done before the array pointed by "out_properties->address" is read or written. The data in the ROM does not have to be flushed. When you want to flush, flush CPU cache directly, call "R_GRAPHICS_Finish" or sandwich reading or writing the image data between "R_GRAPHICS_BeginSoftwareRendering" and "R_GRAPHICS_EndSoftwareRendering".

Argument	graphics_image _t* self	Image
	graphics_image_properties _t* out_properties	(Output) Image properties (6.6.5.14)
Return value	Error code. If there is no error, the return value is 0.	

6.9.3 Functions Equivalent to graphics_pattern_t Class Member Function

6.9.3.1 List of Functions

Section	Function Name	Description
6.9.3.2	R_GRAPHICS_PATTERN_Initialize	Initializes the pattern object.

6.9.3.2 R_GRAPHICS_PATTERN_Initialize

R_GRAPHICS_PATTERN_Initialize		
Outline	Initializes the pattern object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_PATTERN_Initialize(graphics_pattern_t* self, graphics_image_t* image, repetition_t repetition, graphics_t* context);	
Description	Initializes internal variables. Set the object of GraphicsPatternClass to R_GRAPHICS_SetFillPattern.	
Argument	graphics_pattern_t* self	Pattern object
	graphics_image_t* image	Pattern component image
	repetition_t repetition	Repetition setting (normally GRAPHICS_REPEAT)
	graphics_t* context	Belonging context
Return value	Error code. If there is no error, the return value is 0.	

6.9.4 Functions Related to Creating C++ Object

6.9.4.1 List of Functions

Section	Function Name	Description
6.9.4.2	R_RGA_New_Canvas2D_ContextClass	Creates an object of Canvas2D_ContextClass.
6.9.4.3	R_RGA_New_Canvas2D_ImageClass	Creates an object of Canvas2D_ImageClass

6.9.4.2 R_RGA_New_Canvas2D_ContextClass

R_RGA_New_Canvas2D_ContextClass	
Outline	Creates an object of Canvas2D_ContextClass.
Header	RGA.h
Declaration	Canvas2D_ContextClass R_RGA_New_Canvas2D_ContextClass(Canvas2D_ContextConfigClass& in_out_Config); Canvas2D_ContextClass R_RGA_New_Canvas2D_ContextClass(graphics_config_t* in_out_config);
Description	Initializes internal variables. When the object is not used, call the destroy method.
Argument	Canvas2D_ContextConfigClass& in_out_Config See section 6.6.4.3 graphics_config_t* in_out_config See section 6.6.5.2.
Return value	Canvas2D context object. Error = undefined

6.9.4.3 R_RGA_New_Canvas2D_ImageClass

R_RGA_New_Canvas2D_ImageClass	
Outline	Creates an object of Canvas2D_ImageClass
Header	RGA.h
Declaration	Canvas2D_ImageClass R_RGA_New_Canvas2D_ImageClass();
Description	This function is corresponding to new Image() of Canvas2D.
Argument	
Return value	Created an object as Canvas2D_ImageClass. Error = undefined

6.9.5 Canvas2D_ContextClass Member Functions

6.9.5.1 List of Functions

Section	Function Name	Description
6.9.5.2	destroy	Deletes the object of Canvas2D_ContextClass.
6.9.5.3	clearRect	Clears the rectangle area.
6.9.5.4	save	Saves the set value of context in the internal stack.
6.9.5.5	restore	Returns the set value of context to the context from the internal stack.
6.9.5.6	drawImage	Draws an image.
6.9.5.7	createImageData	Creates an image object of r8g8b8a8_t.
6.9.5.8	getImageData	Creates an image to which a part of frame buffer being displayed is copied.
6.9.5.9	putImageData	Draws an image.
6.9.5.10	fillRect	Fill the rectangle area.
6.9.5.11	createPattern	Creates a pattern object.
6.9.5.12	beginPath	Resets the default path content to null.
6.9.5.13	rect	Adds a rectangle to the default path.
6.9.5.14	clip	Sets the shape of the current default path to a clipping area.
6.9.5.15	setTransform	Sets each element of the matrix.
6.9.5.16	translate	Translates the current matrix.

6.9.5.17	scale	Enlarges or reduces the current matrix.
6.9.5.18	rotate	Rotates the current matrix. Rotation center: (0,0)
6.9.5.19	transform	Multiplies the specified 2x3 matrix by the current matrix.

6.9.5.2 destroy (Canvas2D_ContextClass)

destroy (Canvas2D_ContextClass)	
Outline	Deletes the object of Canvas2D_ContextClass.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::destroy();
Description	
Argument	
Return value	

6.9.5.3 clearRect (Canvas2D_ContextClass)

clearRect (Canvas2D_ContextClass)		
Outline	Clears the rectangle area.	
Header	RGA.h	
Declaration	void Canvas2D_ContextClass::clearRect(int_t MinX, int_t MinY, int_t Width, int_t Height);	
Description	Use R_GRAPHICS_SetBackgroundColor() for setting the clear color. Specify the c_LanguageContext (Canvas2D_ContextClass) property for the first argument. Clear color can be gotten by "R_GRAPHICS_GetClearColor" function. Since the drawing buffer content is cleared, cleared content is not displayed only by calling this function. To apply the displayed content, use R_WINDOW_SURFACES_SwapBuffers(). This function is affected by clipping.	
Argument	int_t MinX, int_t MinY	Minimum X and Y coordinates of rectangle
	int_t Width, int_t Height	Width and height of rectangle
Return value		

6.9.5.4 save (Canvas2D_ContextClass)

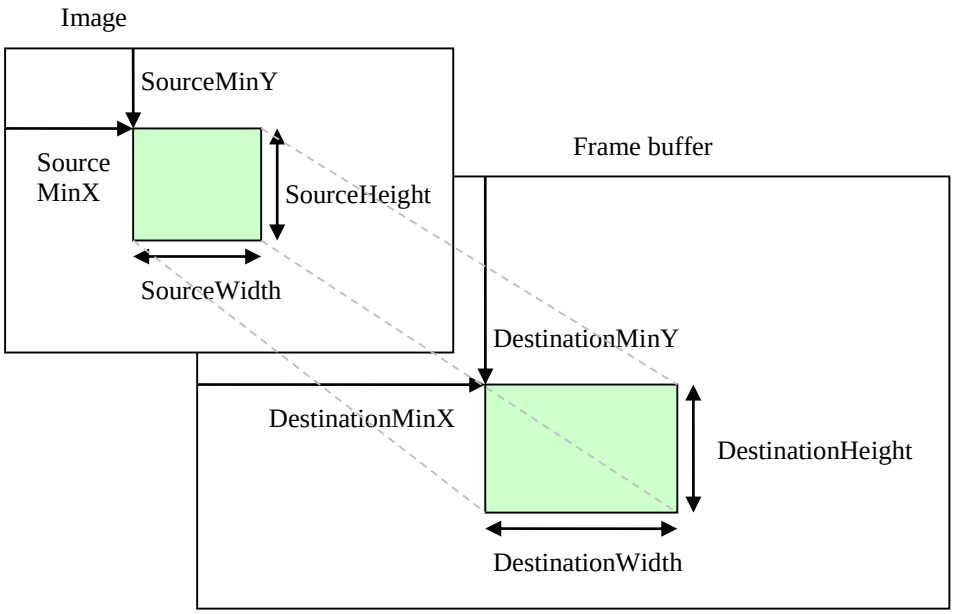
save (Canvas2D_ContextClass)	
Outline	Saves the set value of context in the internal stack.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::save();
Description	A heap area is secured internally for the internal stack.
Argument	
Return value	

6.9.5.5 restore (Canvas2D_ContextClass)

restore (Canvas2D_ContextClass)	
Outline	Returns the set value of context to the context from the internal stack.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::restore();
Description	Releases the heap area used internally as an internal stack.
Argument	
Return value	Error code. If there is no error, the return value is 0.

6.9.5.6 drawImage (Canvas2D_ContextClass)

drawImage (Canvas2D_ContextClass)	
Outline	Draws an image.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::drawImage(graphics_image_t* Image, int_t MinX, int_t MinY); void Canvas2D_ContextClass::drawImage(graphics_image_t* Image, int_t MinX, int_t MinY, int_t Width, int_t Height); void Canvas2D_ContextClass::drawImage(graphics_image_t* Image, int_t SourceMinX, int_t SourceMinY, int_t SourceWidth, int_t SourceHeight, int_t DestinationMinX, int_t DestinationMinY, int_t DestinationWidth, int_t DestinationHeight);
Description	JPEG data can be directly specified for arguments. See section 6.11.2, Identifying Image Format. When neither width nor height is specified, the width and height of the drawing destination are the same as the image width and height. When SourceMinX to DestinationHeight are specified, a part of an image is drawn.



When SourceWidth ≠ DestinationWidth or SourceHeight ≠ DestinationHeight, images are enlarged or reduced.
This function is affected by Canvas2D_ContextClass::GlobalAlpha and the current matrix.
When drawing an image in the YUV422 format, if the value converted from MinX or MinY by the matrix is not an even number, an error occurs.
When the alpha component is included in the image and is not included in the drawing target frame buffer, RGB components to be drawn in the frame buffer are blended to values that have been multiplied by the alpha component. When the alpha component is included in the drawing target frame buffer, RGB components are blended to values that have not been multiplied by the alpha component.
An example of pixel format including the alpha component:
ARGB8888, ARGB4444, ARGB1555
An example of pixel format without alpha component:
XRGB8888, RGB565, YUV422

Argument	graphics_image_t* Image	Image
	int_t MinX, int_t MinY	Minimum X and Y coordinates of drawing destination
	int_t Width, int_t Height	Width and height of drawing destination
	int_t SourceMinX, int_t SourceMinY	Minimum X and Y coordinates in the image
	int_t SourceWidth, int_t SourceHeight	Width and height in the image
	int_t DestinationMinX, int_t DestinationMinY	Minimum X and Y coordinates of drawing destination
	int_t DestinationWidth, int_t DestinationHeight	Width and height of drawing destination
Return value		

6.9.5.7 createImageData (Canvas2D_ContextClass)

createImageData (Canvas2D_ContextClass)		
Outline	Creates an image object of r8g8b8a8_t.	
Header	RGA.h	
Declaration	Canvas2D_ImageClass Canvas2D_ContextClass::createImageData(int_t Width, int_t Height);	
	Canvas2D_ImageClass Canvas2D_ContextClass::createImageData(Canvas2D_ImageClass ImageReferencedWidthHeight);	
Description	Secures a memory area internally from the heap area.	
Argument	int_t Width, int_t Height	Width and height of image
	Canvas2D_ImageClass ImageReferencedWidthHeight	Image object that references width and height
Return value	Created Image object. undefined = Error	

6.9.5.8 getImageData (Canvas2D_ContextClass)

getImageData (Canvas2D_ContextClass)		
Outline	Creates an image object to which a part of the frame buffer being displayed is copied.	
Header	RGA.h	
Declaration	Canvas2D_ImageClass Canvas2D_ContextClass::getImageData(int_t MinX, int_t MinY, int_t Width, int_t Height);	
Description		
Argument	int_t MinX, int_t MinY	Minimum X and Y coordinates of range to be acquired (frame buffer coordinates)
	int_t Width, int_t Height	Width and height of range to be acquired
Return value	Created Image object. undefined = Error	

6.9.5.9 putImageData (Canvas2D_ContextClass)

putImageData (Canvas2D_ContextClass)		
Outline	Draws an image.	
Header	RGA.h	
Declaration	void Canvas2D_ContextClass::putImageData(Canvas2D_ImageClass ImageData, int_t MinX, int_t MinY);	
	void Canvas2D_ContextClass::putImageData(Canvas2D_ImageClass ImageData, int_t MinX, int_t MinY, int_t DirtyX, int_t DirtyY, int_t DirtyWidth, int_t DirtyHeight);	
Description	When DirtyX, DirtyY, DirtyWidth, or DirtyHeight is specified, a part of the Image object image is drawn, but the image is not enlarged or reduced.	

Argument	Canvas2D_ImageClass ImageData	Image object containing the image to be drawn
	int_t MinX, int_t MinY	Minimum X and Y coordinates of drawing destination (canvas coordinates)
	int_t DirtyX, int_t DirtyY	Minimum X and Y coordinates in Image (Image coordinates)
	int_t DirtyWidth, int_t DirtyHeight	Size in Image = Size to be drawn
Return value	None	

6.9.5.10 fillRect (Canvas2D_ContextClass)

fillRect (Canvas2D_ContextClass)		
Outline	Fill the rectangle area.	
Header	RGA.h	
Declaration	void Canvas2D_ContextClass::fillRect(int_t MinX, int_t MinY, int_t Width, int_t Height);	
Description	This function is affected by the current matrix and paint of the current fill. No border is drawn. This function is affected by the globalAlpha property.	
Argument	int_t MinX, int_t MinY	Minimum X and Y coordinates of rectangle
	int_t Width, int_t Height	Width and height of rectangle
Return value		

6.9.5.11 createPattern (Canvas2D_ContextClass)

createPattern (Canvas2D_ContextClass)		
Outline	Creates a pattern object.	
Header	RGA.h	
Declaration	Canvas2D_PatternClass Canvas2D_ContextClass::createPattern(graphics_image_t* Image, char* Repetition);	
Description	Set the pattern object for the fillStyle property.	
Argument	graphics_image_t* Image	Image as a pattern component
	char* Repetition	Repetition setting. Specify "repeat."
Return value	Created pattern object	

6.9.5.12 beginPath (Canvas2D_ContextClass)

beginPath (Canvas2D_ContextClass)	
Outline	Resets the default path content to null.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::beginPath();
Description	
Argument	None
Return value	None

6.9.5.13 rect (Canvas2D_ContextClass)

rect (Canvas2D_ContextClass)	
Outline	Adds a rectangle to the default path.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::rect(int_t MinX, int_t MinY, int_t Width, int_t Height);

Description		
Argument	int_t MinX, int_t MinY	Minimum X and Y coordinates of rectangle
	int_t Width, int_t Height	Width and height of rectangle
Return value	None	

6.9.5.14 clip (Canvas2D_ContextClass)

clip (Canvas2D_ContextClass)	
Outline	Sets the shape of the current default path to a clipping area.
Header	RGA.h
Declaration	void Canvas2D_ContextClass::clip();
Description	When the current default path is empty or is not a single rectangle, an error occurs. When the current default path is empty, drawing is disabled in any area. If this function is called when the current clipping area is a part of the frame buffer, the area common to the clipping area and the default path is a new clipping area. To restore entire drawing, call "restore."
Argument	
Return value	None

6.9.5.15 setTransform (Canvas2D_ContextClass)

setTransform (Canvas2D_ContextClass)		
Outline	Sets each element of the current matrix.	
Header	RGA.h	
Declaration	void Canvas2D_ContextClass::setTransform(graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty);	
Description		
Argument	graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty	2x3 matrix $\begin{pmatrix} sx & kx & tx \\ ky & sy & ty \\ 0 & 0 & 1 \end{pmatrix}$
Return value	None	

6.9.5.16 translate (Canvas2D_ContextClass)

translate (Canvas2D_ContextClass)		
Outline	Translates the current matrix (M).	
Header	RGA.h	
Declaration	void Canvas2D_ContextClass::translate(graphics_matrix_float_t tx, graphics_matrix_float_t ty);	
Description	$M = M \cdot \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$	
Argument	graphics_matrix_float_t tx, graphics_matrix_float_t ty	Displacement (When the origin is at the upper left, plus of X means right direction and plus of Y means downward direction.)
Return value	None	

6.9.5.17 scale (Canvas2D_ContextClass)

scale (Canvas2D_ContextClass)	
Outline	Enlarges or reduces the current matrix (M).
Header	RGA.h
Declaration	void Canvas2D_ContextClass::translate(graphics_matrix_float_t sx, graphics_matrix_float_t sy);
Description	$M=M\cdot\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Argument	graphics_matrix_float_t tx, graphics_matrix_float_t ty — Magnification. Center of enlargement/reduction: Origin
Return value	None

6.9.5.18 rotate (Canvas2D_ContextClass)

rotate (Canvas2D_ContextClass)	
Outline	Rotates the current matrix (M). Center coordinates of rotation: (0,0)
Header	RGA.h
Declaration	void Canvas2D_ContextClass::rotate(graphics_matrix_float_t angle);
Description	$M=M\cdot\begin{pmatrix} \cos(angle) & -\sin(angle) & 0 \\ \sin(angle) & \cos(angle) & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Argument	graphics_matrix_float_t angle — Rotation angle (unit: degree). When the origin is at the upper left, plus means clockwise direction.
Return value	None

6.9.5.19 transform (Canvas2D_ContextClass)

transform (Canvas2D_ContextClass)	
Outline	Multiplies the specified 2x3 matrix by the current matrix (M).
Header	RGA.h
Declaration	void Canvas2D_ContextClass::transform(graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty);
Description	$M = M \cdot \begin{pmatrix} sx & kx & tx \\ ky & sy & ty \\ 0 & 0 & 1 \end{pmatrix}$
Argument	graphics_matrix_float_t sx A 2x3 matrix to be multiplied graphics_matrix_float_t ky graphics_matrix_float_t kx graphics_matrix_float_t sy graphics_matrix_float_t tx graphics_matrix_float_t ty
Return value	None

6.9.6 Functions Related to Canvas2D_ImageClass

6.9.6.1 List of Functions

Section	Function Name	Description
6.9.6.2	destroy	Deletes the Canvas2D_ImageClass object.

6.9.6.2 destroy (Canvas2D_ImageClass)

destroy (Canvas2D_ImageClass)	
Outline	Deletes the Canvas2D_ImageClass object.
Header	RGA.h
Declaration	void Canvas2D_ImageClass::destroy();
Description	
Argument	
Return value	None

6.9.7 Functions Related to Canvas2D_PatternClass

6.9.7.1 List of Functions

Section	Function Name	Description
6.9.7.2	destroy	Deletes the Canvas2D_PatternClass object.

6.9.7.2 destroy (Canvas2D_PatternClass)

destroy (Canvas2D_PatternClass)	
Outline	Deletes the Canvas2D_PatternClass object.
Header	RGA.h
Declaration	void Canvas2D_PatternClass::destroy();
Description	
Argument	
Return value	None

6.9.8 Functions Related to WindowSurfacesClass

6.9.8.1 List of Functions

Section	Function Name	Description
6.9.8.2	initialize	Initialize an object as WindowSurfacesClass
6.9.8.3	destroy	Destroy an object as WindowSurfacesClass
6.9.8.4	get_layer_frame_buffer	Acquires the pointer to the frame buffer structure of the specified layer.
6.9.8.5	swap_buffers	Swaps the buffer of the specified layer and displays the drawn content.
6.9.8.6	alloc_offscreen_stack	Allocates the off-screen buffer from VRAM stack.
6.9.8.7	free_offscreen_stack	Frees the off-screen buffer to VRAM stack.
6.9.8.8	do_message_loop	Enters the message loop.
6.9.8.9	access_layer_attributes	Accesses attributes of the specified display layer.

6.9.8.2 initialize (WindowSurfacesClass)

initialize (WindowSurfacesClass)	
Outline	Initialize an object as WindowSurfacesClass
Header	RGA.h
Declaration	errnum_t WindowSurfacesClass::initialize(WindowSurfacesConfigClass& in_out_config);
Description	See (6.9.9.3) R_WINDOW_SURFACES_Initialize
Argument	WindowSurfacesConfigClass& in_out_config — Parameters. See 6.6.4.7.
Return value	Error code. If there is no error, the return value is 0.

6.9.8.3 destroy (WindowSurfacesClass)

destroy (WindowSurfacesClass)	
Outline	Destroy an object as WindowSurfacesClass
Header	RGA.h
Declaration	void WindowSurfacesClass::destroy();
Description	See (6.9.9.4) R_WINDOW_SURFACES_Finalize
Argument	None
Return value	None

6.9.8.4 get_layer_frame_buffer (WindowSurfacesClass)

get_layer_frame_buffer (WindowSurfacesClass)		
Outline	Acquires the pointer to the frame buffer structure of the specified layer.	
Header	RGA.h	
Declaration	errnum_t WindowSurfacesClass::get_layer_frame_buffer(int_fast32_t layer_num, frame_buffer_t** out_frame_buffer);	
Description	See (6.9.9.5) R_WINDOW_SURFACES_GetLayerFrameBuffer	
Argument	int_fast32_t layer_num	Layer number 0: Innermost, +1: Next to the innermost layer
Outline	frame_buffer_t** out_frame_buffer	(Output) Frame buffer structure
Return value	Error code. If there is no error, the return value is 0.	

6.9.8.5 swap_buffers (WindowSurfacesClass)

swap_buffers (WindowSurfacesClass)	
Outline	Swaps the buffer of the specified layer and displays the drawn content.
Header	RGA.h
Declaration	errnum_t WindowSurfacesClass::swap_buffers(int_fast32_t layer_num, Canvas2D_ContextClass& context);
Description	See (6.9.9.7) R_WINDOW_SURFACES_SwapBuffers
Argument	int_fast32_t layer_num Layer number 0: Innermost, +1: Next to the innermost layer
	Canvas2D_ContextClass& context Drawn graphics context
Return value	Error code. If there is no error, the return value is 0.

6.9.8.6 alloc_offscreen_stack (WindowSurfacesClass)

alloc_offscreen_stack (WindowSurfacesClass)	
Outline	Allocates the off-screen buffer from VRAM stack.
Header	RGA.h
Declaration	errnum_t WindowSurfacesClass::alloc_offscreen_stack(frame_buffer_t* in_out_frame_buffer);
Description	See (6.9.9.12) R_WINDOW_SURFACES_AllocOffscreenStack
Argument	frame_buffer_t* in_out_frame_buffer (Input/output) The off-screen buffer
Return value	Error code. If there is no error, the return value is 0.

6.9.8.7 free_offscreen_stack (WindowSurfacesClass)

free_offscreen_stack (WindowSurfacesClass)	
Outline	Frees the off-screen buffer to VRAM stack.
Header	RGA.h
Declaration	errnum_t WindowSurfacesClass::free_offscreen_stack(const frame_buffer_t* frame_buffer);
Description	See (6.9.9.13) R_WINDOW_SURFACES_FreeOffscreenStack
Argument	frame_buffer_t** out_frame_buffer (Input/output) The freeing off-screen buffer
Return value	Error code. If there is no error, the return value is 0.

6.9.8.8 do_message_loop (WindowSurfacesClass)

do_message_loop (WindowSurfacesClass)	
Outline	Enters the message loop.
Header	RGA.h
Declaration	errnum_t WindowSurfacesClass::do_message_loop();
Description	See (6.9.9.8) R_WINDOW_SURFACES_DoMessageLoop
Argument	None
Return value	Error code. If there is no error, the return value is 0.

6.9.8.9 access_layer_attributes (WindowSurfacesClass)

access_layer_attributes (WindowSurfacesClass)	
Outline	Accesses attributes of the specified display layer.
Header	RGA.h

Declaration	errnum_t WindowSurfacesClass::access_layer_attributes(LayerAttributesClass& in_out_Attributes);
Description	See (6.9.9.11) R_WINDOW_SURFACES_AccessLayerAttributes
Argument	LayerAttributesClass& in_out_Attributes (Input/output) Attributes of layer. See 6.6.4.8. Set read or write to "access" member variable.
Return value	Error code. If there is no error, the return value is 0.

6.9.9 Functions Equivalent to window_surfaces_t Class Member Functions

6.9.9.1 List of Functions

Section	Function Name	Description
6.9.9.2	R_WINDOW_SURFACES_InitConst	Initializes internal variables with constants.
6.9.9.3	R_WINDOW_SURFACES_Initialize	Initializes the display device or window and starts displaying graphics.
6.9.9.4	R_WINDOW_SURFACES_Finalize	Finalizes the display device.
6.9.9.5	R_WINDOW_SURFACES_GetLayerFrameBuffer	Acquires the pointer to the frame buffer structure of the specified layer.
6.9.9.6	R_WINDOW_SURFACES_GetLayerCount	Acquires the number of layers.
6.9.9.7	R_WINDOW_SURFACES_SwapBuffers	Swaps the buffer of the specified layer and displays the drawn content.
6.9.9.8	R_WINDOW_SURFACES_SwapBuffersStart	Starts to swap for displaying the drawn content.
6.9.9.9	R_WINDOW_SURFACES_WaitForVSync	Waits for V-Sync signal of the screen.
6.9.9.10	R_WINDOW_SURFACES_DoMessageLoop	Enters the message loop.
6.9.9.11	R_WINDOW_SURFACES_AccessLayerAttributes	Accesses attributes of the specified display layer.
6.9.9.12	R_WINDOW_SURFACES_AllocOffscreenStack	Allocates the off-screen buffer from VRAM stack.
6.9.9.13	R_WINDOW_SURFACES_FreeOffscreenStack	Frees the off-screen buffer to VRAM stack.

6.9.9.2 R_WINDOW_SURFACES_InitConst

R_WINDOW_SURFACES_InitConst		
Outline	Initializes internal variables with constants.	
Header	RGA.h, window_surfaces.h	
Declaration	void R_WINDOW_SURFACES_InitConst(window_surfaces_t* self);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
Return value	None	

6.9.9.3 R_WINDOW_SURFACES_Initialize

R_WINDOW_SURFACES_Initialize		
Outline	Initializes the display device or window.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_Initialize(window_surfaces_t* self, window_surfaces_config_t* in_out_config);	
Description	To control display independently, directly use the frame_buffer_t structure instead of the window_surfaces_t class. Initializes internal variables. The entire screen becomes black after initialization. After calling "R_WINDOW_SURFACES_SwapBuffers" function, to show is started.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	window_surfaces_config_t* in_out_config	See (6.6.5.6).
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.4 R_WINDOW_SURFACES_Finalize

R_WINDOW_SURFACES_Finalize		
Outline	Finalizes the display device.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_Finalize (window_surfaces_t* self, errnum_t e);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code or e, 0 = successful and e = 0	

6.9.9.5 R_WINDOW_SURFACES_GetLayerFrameBuffer

R_WINDOW_SURFACES_GetLayerFrameBuffer		
Outline	Acquires the pointer to the frame buffer structure of the specified layer.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_GetLayerFrameBuffer(window_surfaces_t* self, int_t layer_num, frame_buffer_t** out_frame_buffer);	
Description	See 6.11.6 Layer structure of sample screen controller. When the attribute of frame buffer is changed, call "R_WINDOW_SURFACES_AccessLayerAttributes" function.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t layer_num	The layer number 0: Innermost, +1: Next to the innermost layer
	frame_buffer_t** out_frame_buffer	(Output) Frame buffer structure
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.6 R_WINDOW_SURFACES_GetLayerCount

R_WINDOW_SURFACES_GetLayerCount		
Outline	Acquires the number of layers.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_GetLayerCount(window_surfaces_t* self, int_t* out_layer_count);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t* out_layer_count	(Output) Number of layers
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.7 R_WINDOW_SURFACES_SwapBuffers

R_WINDOW_SURFACES_SwapBuffers	
Outline	Swaps the buffer of the specified layer and displays the drawn content.
Header	RGA.h, window_surfaces.h
Declaration	errnum_t R_WINDOW_SURFACES_SwapBuffers(window_surfaces_t* self, int_t layer_num, graphics_t graphics);
Description	If buffer was single buffer, buffer is not swapped. The drawn content is displayed without calling this function, but the progress of the drawing is displayed instead. If buffer was double buffer, the content after swapping is front of 2 frames. Call "R_GRAPHICS_Clear" function or call "R_GRAPHICS_IMAGE_InitByShareFrameBuffer" function with showing frame buffer and copy front of 1 frame by calling "R_GRAPHICS_DrawImage" function.

	When Graphics = NULL is specified, completion of drawing is not waited before the buffer is swapped.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t layer_num	The layer number 0: Innermost, +1: Next to the innermost layer
	graphics_t graphics	The drawn graphics context, NULL enabled
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.8 R_WINDOW_SURFACES_SwapBuffersStart

R_WINDOW_SURFACES_SwapBuffersStart		
Outline	Starts to swap for displaying the drawn content.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_SwapBuffersStart(window_surfaces_t* self, int_t layer_num, graphics_t graphics, r_ospl_async_t* async);	
Description	Wait for finish drawing by calling "R_GRAPHICS_FinishStart" function before calling this function. See R_DRIVER_TransferStart function - OS Porting Layer "OSPL" Sample Program.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t layer_num	The layer number 0: Innermost, +1: Next to the innermost layer
	graphics_t graphics	The drawn graphics context, NULL enabled
	r_ospl_async_t* async	(Input/Output) Notify setting. NULL is not permitted
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.9 R_WINDOW_SURFACES_WaitForVSync

R_WINDOW_SURFACES_WaitForVSync		
Outline	Waits for V-Sync signal of the screen.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_WaitForVSync(window_surfaces_t* self, int_fast32_t swap_interval, bool_t is_1_v_sync_at_minimum);	
Description	Waits until the V-Sync interrupt enters for the number of times specified by SwapInterval from the previous swap. In the case of Is1VSyncAtMinimum = false, when the V-Sync interrupt has already entered for the number of times specified by SwapInterval, the processing immediately returns from this function.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_fast32_t swap_interval	Number of V-Sync interrupts until the frame buffer is swapped. 0 or more than 0.
	bool_t is_1_v_sync_at_minimum	Information on whether at least one V-Sync is waited. false=never wait. true=wait 1 times or more than 1.
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.10 R_WINDOW_SURFACES_DoMessageLoop

R_WINDOW_SURFACES_DoMessageLoop		
Outline	Enters the message loop.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_DoMessageLoop(window_surfaces_t* self);	
Description	Upon completion of the application, the processing returns from this function. For the terminating method, see the sub-class specifications.	
Argument	window_surfaces_t* self	Frame buffers and screen display
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.11 R_WINDOW_SURFACES_AccessLayerAttributes

R_WINDOW_SURFACES_AccessLayerAttributes		
Outline	Accesses attributes of the specified display layer.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_AccessLayerAttributes(window_surfaces_t* self, layer_attributes_t* in_out_attributes);	
Description	Not all attributes are available. It depends on the device and support status.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	layer_attributes_t* in_out_attributes	(Input/output) Attributes of layer (6.6.5.7)
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.12 R_WINDOW_SURFACES_AllocOffscreenStack

R_WINDOW_SURFACES_AllocOffscreenStack		
Outline	Allocates the off-screen buffer from VRAM stack.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_AllocOffscreenStack(window_surfaces_t* self, frame_bufer_t* in_out_frame_buffer);	
Description	Input member variable in "frame_buffer_t" is "stride", "height", "buffer_count". Output member variable in "frame_buffer_t" is all element of "buffer_address" array. If the memory was few, E_FEW_ARRAY error is returned. Allocated off-screen buffer is freed by calling "R_WINDOW_SURFACES_Finalize" function.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	frame_bufer_t* in_out_frame_buffer	(Input/output) The off-screen buffer
Return value	Error code. If there is no error, the return value is 0.	

6.9.9.13 R_WINDOW_SURFACES_FreeOffscreenStack

R_WINDOW_SURFACES_FreeOffscreenStack		
Outline	Frees the off-screen buffer to VRAM stack.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_FreeOffscreenStack(window_surfaces_t* self, frame_bufer_t* frame_buffer);	
Description	Input member variable in "frame_buffer_t" is "buffer_count", "buffer_address". If freeing order is not reverse of allocating order, E_ACCESS_DENIED error is returned.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	frame_bufer_t* in_out_frame_buffer	(Input/output) The freeing off-screen buffer
Return value	Error code. If there is no error, the return value is 0.	

6.9.10 Functions Related to byte_per_pixel_t Class

6.9.10.1 List of Functions

Section	Function Name	Description
6.9.10.2	R_RGA_BitPerPixelType_To_BytePerPixelType	Converts the number of bits per pixel to the number of bytes per pixel (with decimal part).
6.9.10.3	R_RGA_BytePerPixelType_To_BitPerPixelType	Converts the number of bytes per pixel (with decimal part) to the number of bits per pixel.
6.9.10.4	R_BYTE_PER_PIXEL_IsInteger	Returns information on whether the number of bytes per pixel is an integer.

6.9.10.2 R_RGA_BitPerPixelType_To_BytePerPixelType

R_RGA_BitPerPixelType_To_BytePerPixelType	
Outline	Converts the number of bits per pixel to the number of bytes per pixel (with decimal part).
Header	RGA.h
Declaration	byte_per_pixel_t R_RGA_BitPerPixelType_To_BytePerPixelType(int_t bit_per_pixel);
Description	
Argument	int_t bit_per_pixel Number of bits per pixel
Return value	Number of bytes per pixel (with decimal part)

6.9.10.3 R_RGA_BytePerPixelType_To_BitPerPixelType

R_RGA_BytePerPixelType_To_BitPerPixelType	
Outline	Converts the number of bytes per pixel (with decimal part) to the number of bits per pixel.
Header	RGA.h
Declaration	int_t R_RGA_BytePerPixelType_To_BitPerPixelType(byte_per_pixel_t byte_per_pixel);
Description	
Argument	byte_per_pixel_t byte_per_pixel Number of bytes per pixel (with decimal part)
Return value	Number of bits per pixel

6.9.10.4 R_BYTE_PER_PIXEL_IsInteger

R_BYTE_PER_PIXEL_isInteger	
Outline	Returns information on whether the number of bytes per pixel is an integer.
Header	RGA.h
Declaration	bool_t R_BYTE_PER_PIXEL_isInteger(byte_per_pixel_t byte_per_pixel);
Description	
Argument	byte_per_pixel_t byte_per_pixel Number of bytes per pixel (with decimal part)
Return value	Information on whether the number of bits per pixel is an integer

6.9.11 Functions Related to v_sync_t Class

6.9.11.1 List of Functions

Section	Function Name	Description
6.9.11.2	R_V_SYNC_Initialize	Attaches to the V-Sync interrupt.
6.9.11.3	R_V_SYNC_Finalize	Detaches from the V-Sync interrupt.
6.9.11.4	R_V_SYNC_Wait	Waits until the V-Sync interrupt enters.
6.9.11.5	R_V_SYNC_WaitStart	Starts to wait for V-Sync interrupt.
6.9.11.6	R_V_SYNC_OnInterrupting	Receives V-Sync interrupt.
6.9.11.7	R_V_SYNC_GetAsyncStatus	Gets the status of asynchronous operation and interrupt.

6.9.11.2 R_V_SYNC_Initialize

R_V_SYNC_Initialize	
Outline	Attaches to the V-Sync interrupt.
Header	RGA.h.h, vsync.h
Declaration	errnum_t R_V_SYNC_Initialize(int_fast32_t ChannelNum);
Description	Calls back the porting layer function of the interrupt from inside of this function. When the V-Sync interrupt has already been attached, an error occurs. When the V-Sync interrupt is controlled by the application, this function cannot be used. This function is used in the window_surfaces_t class. The v_sync_t class cannot be used when the window_surfaces_t class is used.
Argument	int_fast32_t ChannelNum Channel number of the screen
Return value	Error code. If there is no error, the return value is 0.

6.9.11.3 R_V_SYNC_Finalize

R_V_SYNC_Finalize	
Outline	Detaches from the V-Sync interrupt.
Header	RGA.h.h, vsync.h
Declaration	errnum_t R_V_SYNC_Finalize(int_fast32_t ChannelNum, errnum_t e);
Description	
Argument	int_fast32_t ChannelNum Channel number of the screen
Return value	Error code or e, 0 = successful and e = 0

6.9.11.4 R_V_SYNC_Wait

R_V_SYNC_Wait		
Outline	Waits until the V-Sync interrupt enters.	
Header	RGA.h.h, vsync.h	
Declaration	errnum_t R_V_SYNC_WaitForInterrupt(int_fast32_t ChannelNum, int_fast32_t SwapInterval, bool_t Is1VSyncAtMinimum);	
Description	Waits until the V-Sync interrupt enters for the number of times specified by SwapInterval from the previous swap. In the case of Is1VSyncAtMinimum = false, when the V-Sync interrupt has already entered for the number of times specified by SwapInterval, the processing immediately returns from this function.	
Argument	int_fast32_t ChannelNum	Channel number of the screen
	int_fast32_t SwapInterval	Number of V-Sync interrupts until the frame buffer is swapped. 0 or more than 0.
	bool_t Is1VSyncAtMinimum	Information on whether at least one V-Sync is waited. false=never wait. true=wait 1 times or more than 1.
Return value	Error code. If there is no error, the return value is 0.	

6.9.11.5 R_V_SYNC_WaitStart

R_V_SYNC_WaitStart		
Outline	Starts to wait for V-Sync interrupt.	
Header	RGA.h, vsync.h	
Declaration	errnum_t R_V_SYNC_WaitStart(int_fast32_t ChannelNum, int_fast32_t SwapInterval, bool_t Is1VSyncAtMinimum, r_ospl_async_t* Async);	
Description	This is the asynchronous function of R_V_SYNC_Wait function. See R_DRIVER_TransferStart function - OS Porting Layer "OSPL" Sample Program.	
Argument	int_fast32_t ChannelNum	Channel number of the screen
	int_fast32_t SwapInterval	Number of V-Sync interrupts until the frame buffer is swapped. 0 or more than 0.
	bool_t Is1VSyncAtMinimum	Information on whether at least one V-Sync is waited. false=never wait. true=wait 1 times or more than 1.
	r_ospl_async_t* Async	(Input/Output) Notify setting. NULL is not permitted
Return value	Error code. If there is no error, the return value is 0.	

6.9.11.6 R_V_SYNC_OnInterrupting

R_V_SYNC_OnInterrupting		
Outline	Receives V-Sync interrupt.	
Header	RGA.h, vsync.h	
Declaration	errnum_t R_V_SYNC_OnInterrupting(r_ospl_interrupt_t* InterruptSource);	
Description	See R_DRIVER_OnInterrupting function - OS Porting Layer "OSPL" Sample Program.	
Argument	r_ospl_interrupt_t* InterruptSource	Interrupt source
Return value	Error code. If there is no error, the return value is 0.	

6.9.11.7 R_V_SYNC_GetAsyncStatus

R_V_SYNC_GetAsyncStatus	
Outline	Gets the status of asynchronous operation and interrupt.

Header	RGA.h, vsync.h	
Declaration	errnum_t R_V_SYNC_GetAsyncStatus(int_fast32_t ChannelNum, const r_v_sync_async_status_t** out_Status);	
Description	See R_DRIVER_GetAsyncStatus function - OS Porting Layer "OSPL" Sample Program.	
Argument	int_fast32_t ChannelNum	Channel number of the screen
	const r_v_sync_async_status_t** out_Status	(Output) Pointer to the structure of interrupt and asynchronous operation
Return value	Error code. If there is no error, the return value is 0.	

6.9.12 Functions Related to vram_ex_stack_t class

6.9.12.1 List of Functions

Section	Function Name	Description
6.9.12.2	R_VRAM_EX_STACK_Initialize	Initialize the stack in the external RAM.
6.9.12.3	R_VRAM_EX_STACK_Alloc	Allocates the off-screen buffer from the external RAM.
6.9.12.4	R_VRAM_EX_STACK_Free	Frees the off-screen buffer to the external RAM.

6.9.12.2 R_VRAM_EX_STACK_Initialize

R_VRAM_EX_STACK_Initialize		
Outline	Initialize the stack in the external RAM.	
Header	RGA.h, RGA_SampleLib_Port.h	
Declaration	errnum_t R_VRAM_EX_STACK_Initialize(vram_ex_stack_t* self, void* null_config);	
Description	If to re-initialize was done, all allocated off-screen buffer is freed.	
Argument	vram_ex_stack_t* self	The stack in the external RAM
	void* null_config	The reserved variable. Pass NULL
Return value	Error code. If there is no error, the return value is 0.	

6.9.12.3 R_VRAM_EX_STACK_Alloc

R_VRAM_EX_STACK_Alloc		
Outline	Allocates the off-screen buffer from the external RAM.	
Header	RGA.h, RGA_SampleLib_Port.h	
Declaration	errnum_t R_VRAM_EX_STACK_Alloc(vram_ex_stack_t* self, frame_bufer_t* in_out_frame_buffer);	
Description	Input member variable in "frame_buffer_t" is "stride", "height", "buffer_count". Output member variable in "frame_buffer_t" is all element of "buffer_address" array. If the memory was few, E_FEW_ARRAY error is returned. Allocated off-screen buffer is freed by calling "R_VRAM_EX_STACK_Initialize" function.	
Argument	vram_ex_stack_t* self	The stack in the external RAM
	frame_bufer_t* in_out_frame_buffer	(Input/output) The off-screen buffer
Return value	Error code. If there is no error, the return value is 0.	

6.9.12.4 R_VRAM_EX_STACK_Free

R_VRAM_EX_STACK_Free		
Outline	Frees the off-screen buffer to the external RAM.	
Header	RGA.h, RGA_SampleLib_Port.h	
Declaration	errnum_t R_VRAM_EX_STACK_Free(vram_ex_stack_t* self, frame_bufer_t* frame_buffer);	
Description	Input member variable in "frame_buffer_t" is "buffer_count", "buffer_address". If freeing order is not reverse of allocating order, E_ACCESS_DENIED error is returned.	
Argument	vram_ex_stack_t* self	The stack in the external RAM
	frame_bufer_t* in_out_frame_buffer	(Input/output) The freeing off-screen buffer
Return value	Error code. If there is no error, the return value is 0.	

6.9.13 Functions Related to animation_timing_function_t class

6.9.13.1 List of Functions

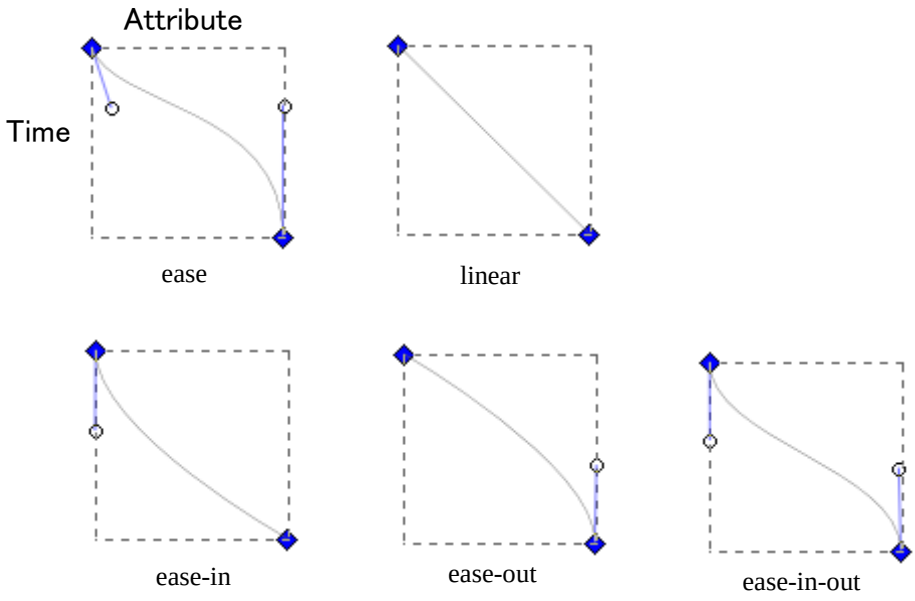
Section	Function Name	Description
6.9.13.2	R_Get_AnimationTimingFunction	Gets defined animation timing.
6.9.13.3	R_ANIMATION_TIMING_FUNCTION_GetValue	Calculates an attribute value at the specified elapsed time from the time starting animation.

6.9.13.2 R_Get_AnimationTimingFunction

R_Get_AnimationTimingFunction		
Outline	Gets defined animation timing.	
Header	RGA.h	
Declaration	errnum_t R_Get_AnimationTimingFunction(char* timing_name, animation_timing_function_t** out_timing);	
Description		
Argument	char* timing_name	The name of animation timing. The following name can be passed: "ease", "linear", "ease_in", "ease_out", "ease_in_out"
	animation_timing_function_t** out_timing	(Output) The address of animation timing object
Return value	Error code. If there is no error, the return value is 0.	

6.9.13.3 R_ANIMATION_TIMING_FUNCTION_GetValue

R_ANIMATION_TIMING_FUNCTION_GetValue	
Outline	Calculates an attribute value at the specified elapsed time from the time starting animation.
Header	RGA.h
Declaration	float32_t R_ANIMATION_TIMING_FUNCTION_GetValue(animation_timing_function_t* self, float32_t clamp_time, float32_t value_of_previous_keyframe, float32_t value_of_next_keyframe);
Description	Attribute value is user defined position value, color value or other value changed by time



Example:

```
timing_name="linear", value_of_previous_keyframe=10,
value_of_next_keyframe=20, clamp_time=0.5
```

This case returns 15.

Argument	animation_timing_function_t* self	The animation timing object
	float32_t clamp_time	The percentage of time from previous key frame (clamp_time=0.0) to next key frame (clamp_time=1.0). (decimal from 0.0 to 1.0)
	float32_t value_of_previous_keyframe	The value of attribute at the previous key frame
	float32_t value_of_next_keyframe	The value of attribute at the next key frame
Return value	The value of attribute at the time of "clamp_time" argument	

6.9.14 Other Functions

6.9.14.1 List of Functions

Section	Function Name	Description
6.9.14.2	R_RGA_Get_R8G8B8A8	Returns the R8G8B8A8 color value.
6.9.14.3	R_RGA_CalcWorkBufferSize	Calculates the size required for the work buffer.
6.9.14.4	R_RGA_CalcWorkBufferB_Size	Calculates the size required for the work buffer B.

6.9.14.2 R_RGA_Get_R8G8B8A8

R_RGA_Get_R8G8B8A8		
Outline	Returns the R8G8B8A8 color value.	
Header	RGA.h	
Declaration	r8g8b8a8_t R_RGA_Get_R8G8B8A8(int_t red, int_t green, int_t blue, int_t alpha);	
Description		
Argument	int_t red, int_t green, int_t blue, int_t alpha	Each color component 0 to 255
Return value	R8G8B8A8 color value	

6.9.14.3 R_RGA_CalcWorkBufferSize

R_RGA_CalcWorkBufferSize		
Outline	Calculates the size required for the work buffer.	
Header	RGA.h	
Declaration	size_t R_RGA_CalcWorkBufferSize(int_t MaxHeightOfFrameBuffer);	
Description	Parameters may change in the future. R_RGA_CalcWorkBufferSize is the #define macro. Requested size for work buffer: Maximum length of display list + 64 * (Maximum height of the frame buffer (multiple of 8)) *4*2 [Bytes] Maximum length of display list = 184 See: 6.6.5.3 graphics_config_t	
Argument	int_t MaxHeightOfFrameBuffer	Maximum height of the frame buffer to be a drawing destination
Return value	Size (bytes) required for the work buffer	

6.9.14.4 R_RGA_CalcWorkBufferB_Size

R_RGA_CalcWorkBufferB_Size		
Outline	Calculates the size required for the work buffer B.	
Header	RGA.h	
Declaration	size_t R_RGA_CalcWorkBufferB_Size(int_t MaxWidthOfJPEG, int_t MaxHeightOfJPEG, int_t MaxBytePerPixelOfFrameBuffer);	
Description	Parameters may change in the future. R_RGA_CalcWorkBufferB_Size is the #define macro. Requested size for work buffer B: ceil_16(MaxWidthOfJPEG) * ceil_16(MaxHeightOfJPEG) * MaxBytePerPixelOfFrameBuffer [Bytes] ceil_16: round up to multiples of 16 When PNG image was decoded, the work area (size is set to work_size_for_libPNG member variable) of LibPNG is allocated in work buffer B. But R_RGA_CalcWorkBufferB_Size macro is not added this size.	

The work buffer B is not requested, if all following condition were fulfilled (JPEG Codec Unit (JCU) in RZ/A1H, M can draw directly) or any JPEG images are not drawn.

- The drawing target address of left up of JPEG image can be divided by 8.
- The size of JPEG image is multiples of MCU (Minimum Coded Unit). The size is depend on the pixel format of the JPEG data.
  - 16pixels x 8lines (JPEG image is YCbCr422 format)
  - 16pixels x 16lines (JPEG image is YCbCr420 format)
  - 8pixels x 8lines (JPEG image is YCbCr444 format)
  - 32pixels x 8lines (JPEG image is YCbCr411 format)
- The matrix is unit matrix or translation only.

The return value is set in "graphics_config_t" type. See 6.6.5.3 graphics_config_t

Argument	int_t MaxWidthOfJPEG	The maximum width of JPEG image
	int_t MaxHeightOfJPEG	The maximum height of JPEG image
	int_t MaxBytePerPixelOfFrameBuffer	The maximum bytes of the drawing target frame buffer. If the matrix is not unit matrix and not translation only, this argument is 4
Return value	Size (bytes) required for the work buffer	

6.9.15 Functions in strings

6.9.15.1 List of Functions

Section	Function Name	Description
6.9.15.2	rgb	Returns the value of color from CSS Color format
6.9.15.3	rgba	Returns the value of color from CSS Color format with alpha

6.9.15.2 rgb

rgb	
Outline	Returns the value of color from CSS Color format
Header	RGA.h
Declaration	r8g8b8a8_t rgb(int_t red_max255, int_t green_max255, int_t blue_max255);
Description	Example: "rgb(255, 255, 0)" Target: fillStyle (6.8.1.3) The value of alpha component is maximum value (=1.0).
Argument	int_t red_max255 The value of red component (0-255) int_t green_max255 The value of green component (0-255) int_t blue_max255 The value of blue component (0-255)
Return value	The value of R8G8B8A8 color.

6.9.15.3 rgba

rgba	
Outline	Returns the value of color from CSS Color format with alpha
Header	RGA.h
Declaration	r8g8b8a8_t rgba(int_t red_max255, int_t green_max255, int_t blue_max255, float32_t alpha_max1);
Description	Example: "rgba(255, 255, 0, 0.5)" Target: fillStyle (6.8.1.3)
Argument	int_t red_max255 The value of red component (0-255) int_t green_max255 The value of green component (0-255) int_t blue_max255 The value of blue component (0-255) float32_t alpha_max1 The value of alpha component (0.0-1.0)
Return value	The value of R8G8B8A8 color.

6.10 Porting layer Functions

This section describes porting layer functions called back from RGA. The package contains a sample which can be modified by the user.

For functions with names that end with _FuncType, a function to be registered must be called at the beginning of the main function, but the function can be modified dynamically.

Other functions must be solved at the time of linkage and cannot be modified dynamically.

6.10.1 Functions on Default Settings of RGA

6.10.1.1 List of Functions

Section	Function Name	Description
6.10.1.2	R_GRAPHICS_STATIC_OnInitializeDefault	Default of the callback function that sets the default graphics_config_t value
6.10.1.3	R_GRAPHICS_OnInitialize_FuncType	Type of the callback function that sets the default graphics_config_t value
6.10.1.4	R_GRAPHICS_STATIC_OnFinalizeDefault	Default of the function that releases memory allocated in the R_GRAPHICS_OnInitialize_FuncType function
6.10.1.5	R_GRAPHICS_OnFinalize_FuncType	Type of the function that releases memory allocated in the R_GRAPHICS_OnInitialize_FuncType function

6.10.1.2 R_GRAPHICS_STATIC_OnInitializeDefault

R_GRAPHICS_STATIC_OnInitializeDefault		
Outline	Default of the callback function that sets the default graphics_config_t value	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_STATIC_OnInitializeDefault(graphics_t* self, graphics_config_t* in_out_config, void** out_default_object);	
Description	Default of the R_GRAPHICS_OnInitialize_FuncType function type	
Argument	graphics_t* self	Address of the object (non-initialized) at which initialization starts
	graphics_config_t* in_out_config	See section 6.6.5.3.
	void** out_default_object	(Output) Memory allocated in this function
Return value	Error code. If there is no error, the return value is 0.	

6.10.1.3 R_GRAPHICS_OnInitialize_FuncType

R_GRAPHICS_OnInitialize_FuncType		
Outline	Type of the callback function that sets the default graphics_config_t value	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_OnInitialize_FuncType(graphics_t* self, graphics_config_t* in_out_config, void** out_default_object);	
Description	Use R_GRAPHICS_STATIC_SetOnInitialize() to register callback functions of this function type. Callback functions of this function type are called back immediately after R_GRAPHICS_Initialize() is called. *out_default_object is used only to be released in inter R_GRAPHICS_OnInitialize_FuncType(). When R_GRAPHICS_OnInitialize_FuncType() is not called back, it is not necessary to set *out_default_object.	
Argument	graphics_t* self	Address of the object (non-initialized) at which initialization starts
	graphics_config_t* in_out_config	See section 6.6.5.3.
	void** out_default_object	(Output) Memory allocated in this function

Return value	Error code. If there is no error, the return value is 0.

6.10.1.4 R_GRAPHICS_STATIC_OnFinalizeDefault

R_GRAPHICS_STATIC_OnFinalizeDefault		
Outline	Default of the function that releases memory allocated in the R_GRAPHICS_OnInitialize_FuncType function	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_STATIC_OnFinalizeDefault(graphics_t* self, void* default_object, errnum_t e);	
Description	Default of the R_GRAPHICS_OnFinalize_FuncType function type	
Argument	graphics_t* self	Address of the object that has been finalized
	void* default_object	(Output) Memory allocated in this function
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code or e, 0 = successful and e = 0	

6.10.1.5 R_GRAPHICS_OnFinalize_FuncType

R_GRAPHICS_OnFinalize_FuncType		
Outline	Type of the function that releases memory allocated in the R_GRAPHICS_OnInitialize_FuncType function	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_OnFinalize_FuncType(graphics_t* self, void* default_object, errnum_t e);	
Description	Use GRAPHICS_STATIC_SetOnFinalize() to register callback functions of this function type. Callback functions of this function type are called back from the end of the R_GRAPHICS_Finalize() function. The output default_object value is contained in *out_default_object of the R_GRAPHICS_OnInitialize_FuncType() function.	
Argument	graphics_t* self	Address of the object that has been finalized
	void* default_object	(Output) Memory allocated in this function
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code or e, 0 = successful and e = 0	

6.10.2 Porting layer functions of OSPL

RGA uses following functions of OSPL API. If the memory map or OS was changed, change define of OSPL API functions or replace to functions already changed. More information is in the document of RZ/A1 group OS Porting Layer "OSPL" Sample Program (R01AN1887EJ).

- Functions depended on the memory map
  - R_OSPL_ToCachedAddress
  - R_OSPL_ToUncachedAddress
  - R_OSPL_ToPhysicalAddress
  - R_OSPL_MEMORY_GetLevelOfFlush
  - R_OSPL_AXI_Get2ndCacheAttribute
- Functions depended on OS
  - R_OSPL_THREAD_GetCurrentId
  - R_OSPL_EVENT_Set
  - R_OSPL_EVENT_Clear
  - R_OSPL_EVENT_Wait
  - R_OSPL_MEMORY_Flush
  - R_OSPL_Delay
- Functions depended on the compiler or OS
  - INLINE
  - STATIC_INLINE
  - R_OSPL_SECTION
  - R_OSPL_ALIGNMENT
  - R_OSPL_EnableAllInterrupt
  - R_OSPL_DisableAllInterrupt
  - R_OSPL_MEMORY_Barrier
- Functions depended on hardware
  - R_OSPL_FTIMER_InitializeIfNot
  - R_OSPL_FTIMER_Get
- Functions no depended on
  - (some functions)

6.10.3 Porting layer functions of RGPNCG

RGA uses following functions of RGPNCG. If the OS was changed, change define of RGPNCG functions or replace functions to function already changed

.

If NCGSYS_*State functions use event flags of OSPL, it is necessary to port NCGSYS_*State functions.

- Functions depended on OS

Section	Function Name	Description
6.10.3.1	NCGSYS_CreateState	Creates an event flags of RGPNCG.
6.10.3.2	NCGSYS_DestroyState	Destroys the event flags of RGPNCG.
6.10.3.3	NCGSYS_SetState	Changes the value of the event flags of RGPNCG.
6.10.3.4	NCGSYS_GetState	Gets the value of the event flags of RGPNCG.
6.10.3.5	NCGSYS_WaitState	Waits for the value of the event flags of RGPNCG until specified value.
6.10.3.6	NCGSYS_SetStateEventValue	Sets using value of OS event flags
6.10.3.7	NCGSYS_GetLastCreatedState	Returns last created event flags
6.10.3.8	NCGVG_Attach_ISR	Registers the interrupt callback function for R-GPVG2.
6.10.3.9	NCGVG_Detach_ISR	Unregisters the interrupt callback function for R-GPVG2.
6.10.3.10	NCGVGISRfp	Type of the interrupt callback function for R-GPVG2.

- Functions depended on the policy of Power-Down Modes

Section	Function Name	Description
6.10.3.11	NCGVG_Init	Starts to use R-GPVG2
6.10.3.12	NCGVG_DeInit	Ends to use R-GPVG2

- Functions no depended on
  - NCGSYS_Abort
  - NCGSYS_CPUVAddrToSysPAddr
  - NCGSYS_ReadReg
  - NCGSYS_WriteReg

6.10.3.1 NCGSYS_CreateState

NCGSYS_CreateState		
Outline	Creates an event flags of RGPNCG.	
Header	ncg_state.h	
Declaration	NCGint32 NCGSYS_CreateState(NCGvoid** ppObj, NCGuint32 ui32StateID);	
Description	Change define of this function for the target OS. If thread attached event of OSPL was called, it is not have to change define.	
Argument	NCGvoid** ppObj	(Output) A event flags object
	NCGuint32 ui32StateID	Not used
Return value	Error code. If there is no error, the return value is NCG_no_err.	

6.10.3.2 NCGSYS_DestroyState

NCGSYS_DestroyState	
Outline	Destroys the event flags of RGPNCG.
Header	ncg_state.h

Declaration	NCGint32 NCGSYS_DestroyState(NCGvoid *pObj);	
	If thread attached event of OSPL was called, it is not have to change define.	
Description	Change define of this function for the target OS.	
Argument	NCGvoid* pObj	The event flags object
Return value	Error code. If there is no error, the return value is NCG_no_err.	

6.10.3.3 NCGSYS_SetState

NCGSYS_SetState		
Outline	Changes the value of the event flags of RGPNCG.	
Header	ncg_state.h	
Declaration	NCGint32 NCGSYS_SetState(NCGvoid* pObj, NCGuint32 ui32State, NCGuint32 ui32Flags);	
Description	Change define of this function for the target OS.	
	If thread attached event of OSPL was called, it is not have to change define.	
Argument	NCGvoid* pObj	The event flags object
	NCGuint32 ui32State	See the description of ui32Flags argument
	NCGuint32 ui32Flags	Case of NCGSYS_STATE_SET_SET: Changes all bits of the event flags to the value of ui32State argument Case of NCGSYS_STATE_SET_OR: Does the OR operation with the event flags. Sets 1 to the event flags' bits of the column set 1 in ui32State argument. Case of NCGSYS_STATE_SET_AND: Does the AND operation with the event flags. Sets 0 to the event flags' bits of the column set 0 in ui32State argument.
Return value	Error code. If there is no error, the return value is NCG_no_err.	

6.10.3.4 NCGSYS_GetState

NCGSYS_GetState		
Outline	Gets the value of the event flags of RGPNCG.	
Header	ncg_state.h	
Declaration	NCGuint32 NCGSYS_GetState(NCGvoid* pObj, NCGuint32 ui32Flags);	
Description	Change define of this function for the target OS.	
	If thread attached event of OSPL was called, it is not have to change define.	
Argument	NCGvoid* pObj	The event flags object
	NCGuint32 ui32Flags	Not used
Return value	The value of the event flags.	

6.10.3.5 NCGSYS_WaitState

NCGSYS_WaitState		
Outline	Waits for the value of the event flags of RGPNCG until specified value.	
Header	ncg_state.h	
Declaration	NCGint32 NCGSYS_WaitState(NCGvoid* pObj, NCGuint32 ui32State, NCGuint32 ui32Flags, NCGuint32 ui32Timeout);	
Description	Change define of this function for the target OS.	
	If thread attached event of OSPL was called, it is not have to change define.	
Argument	NCGvoid* pObj	The event flags object
	NCGuint32 ui32State	The value of bits set to 1 waiting for bits that become to 1

	NCGuint32 ui32Flags	Case of NCGSYS_STATE_WAIT_AND: Waits for all bits are 1 that bits are 1 in the specified ui32State argument. Case of NCGSYS_STATE_WAIT_OR: Waits for any bits are 1 that bits are 1 in the specified ui32State argument.
	NCGuint32 ui32Timeout	NCG_TIMEOUT_INFINITE is passed.
Return value	Error code. If there is no error, the return value is NCG_no_err.	

6.10.3.6 NCGSYS_SetStateEventValue

NCGSYS_SetStateEventValue		
Outline	Sets using value of OS event flags	
Header	ncg_state.h	
Declaration	NCGvoid NCGSYS_SetStateEventValue (NCGvoid* pObj, NCGuint32 ui32EventValue);	
Description	Define this function, if not defined. If thread attached event of OSPL was called, it is not have to change define. Event flags specified with "NCGSYS_SetState" function is set to variables in NCG. Event flags specified with "NCGSYS_SetStateEventValue" function is set to OS objects.	
Argument	NCGvoid* pObj	A event flag of NCG
	NCGuint32 ui32EventValue	Value setting to OS event flags
Return value	None	

6.10.3.7 NCGSYS_GetLastCreatedState

NCGSYS_GetLastCreatedState	
Outline	Returns last created event flags
Header	ncg_state.h
Declaration	NCGvoid* NCGSYS_GetLastCreatedState(void);
Description	Define this function, if not defined. If thread attached event of OSPL was called, it is not have to change define. This function returns an event flags created by "NCGSYS_CreateState" function. Thread preemption does not occurred from calling "NCGSYS_CreateState" function to calling "NCGSYS_GetLastCreatedState" function.
Argument	None
Return value	Last created event flags

6.10.3.8 NCGVG_Attach_ISR

NCGVG_Attach_ISR		
Outline	Registers the interrupt callback function for Renesas Graphics Processor for OpenVG™ (R-GPVG2).	
Header	ncg_vg_isr.h	
Declaration	NCGint32 NCGVG_Attach_ISR(NCGVGISRfp pfnInterrupt);	
Description	Change define of this function for the target OS.	
Argument	NCGVGISRfp pfnInterrupt	The interrupt callback function
Return value	Error code. If there is no error, the return value is NCG_no_err.	

6.10.3.9 NCGVG_Detach_ISR

NCGVG_Detach_ISR

Outline	Unregisters the interrupt callback function for Renesas Graphics Processor for OpenVG™ (R-GPVG2).
Header	ncg_vg_isr.h
Declaration	NCGint32 NCGVG_Detach_ISR(NCGVGISRfp pfnInterrupt);
Description	Change define of this function for the target OS.
Argument	NCGVGISRfp pfnInterrupt The interrupt callback function
Return value	Error code. If there is no error, the return value is NCG_no_err.

6.10.3.10 NCGVGISRfp

NCGVGISRfp	
Outline	Type of the interrupt callback function for Renesas Graphics Processor for OpenVG™ (R-GPVG2).
Header	ncg_defs.h
Declaration	NCGuint32 (*NCGVGISRfp)(void);
Description	Call "R_GRAPHICS_OnInterrupting" function from the last of this callback function. In attached "RGPNCG", "NCGVGISRfp" type function is "NCGVG_RGPVG_ISR".
Argument	None
Return value	None

6.10.3.11 NCGVG_Init

NCGVG_Init	
Outline	Starts to use Renesas Graphics Processor for OpenVG™ (R-GPVG2)
Header	ncg_vg.h
Declaration	NCGvoid NCGVG_Init(PNCGVGINFO pVGInfo);
Description	If the Power-Down Mode of Renesas graphics processor for Renesas Graphics Processor for OpenVG™ (R-GPVG2) was the halted state, change to the running state in this function.
Argument	PNCGVGINFO pVGInfo Not used
Return value	None

6.10.3.12 NCGVG_DeInit

NCGVG_DeInit	
Outline	Ends to use Renesas Graphics Processor for OpenVG™ (R-GPVG2)
Header	ncg_vg.h
Declaration	NCGvoid NCGVG_DeInit(PNCGVGINFO pVGInfo);
Description	If necessary, change the Power-Down Mode of Renesas graphics processor for Renesas Graphics Processor for OpenVG™ (R-GPVG2) to the halted state in this function.
Argument	PNCGVGINFO pVGInfo Not used
Return value	None

6.11 Supplementary Explanation

6.11.1 Correspondence to Canvas 2D and Correspondence to Hardware Acceleration

In the hardware column, ✓: Hardware is used, x: No hardware is used, −: Not applicable

Canvas2D API	RGA - C++ API	RGA - C language API	Hardware
1 Conformance requirements			
CanvasRenderingContext2D interface.	Canvas2D_ContextClass	graphics_t	–
getContext()	R_RGA_New_Canvas2D_ContextClass	R_GRAPHICS_Initialize	–
context.canvas	-	-	
CSS currentColor	-	-	
2 The canvas state			
.save ()	.save()	R_GRAPHICS_Save	–
.restore()	.restore()	R_GRAPHICS_Restore	–
3 Transformations			
.scale(x, y)	.scale(x, y)	R_GRAPHICS_ScaleMatrix	✓
.rotate(angle)	.rotate(angle)	R_GRAPHICS_RotateMatrixRadian	✓
.translate(x, y)	.translate(x, y)	R_GRAPHICS_TranslateMatrix	✓
.transform(a, b, c, d, e, f)	.transform(a, b, c, d, e, f)	R_GRAPHICS_TransformMatrix	✓
.setTransform(a, b, c, d, e, f)	.setTransform(a, b, c, d, e, f)	R_GRAPHICS_SetMatrix_2x3	✓
4 Line styles			
.lineWidth	-	-	
.lineCap, .lineJoin, .miterLimit	-	-	
5 Text styles			
	-	-	
6 Building paths			
.moveTo()	-	-	
.closePath()	-	-	
.lineTo()	-	-	
.quadraticCurveTo()	-	-	
.bezierCurveTo()	-	-	
.arcTo()	-	-	
.arc()	-	-	
.rect()	.rect()	R_GRAPHICS_Rect	✓
7 Fill and stroke styles			
.fillStyle single-color	.fillStyle	R_GRAPHICS_SetFillColor	✓
.fillStyle gradation	-	-	
.fillStyle pattern	.fillStyle	R_GRAPHICS_SetFillPattern	✓
.strokeStyle single-color	-	-	
.strokeStyle gradation	-	-	

.strokeStyle pattern	-	-	
.createLinearGradient()	-	-	
.createRadialGradient()	-	-	
CanvasGradient.addColorStop()	-	-	
.createPattern()	.createPattern()	R_GRAPHICS_PATTERN_Initialize	–
8 The current default path			
.beginPath()	.beginPath()	R_GRAPHICS_BeginPath	–
.fill()	-	-	
.stroke()	-	-	
.drawSystemFocusRing()	-	-	
.drawCustomFocusRing()	-	-	
.scrollPathIntoView()	-	-	
.clip()	.clip()	R_GRAPHICS_Clip	✓
	(Only a single rectangle)	(Only a single rectangle)	
.isPointInPath()	-	-	
9 Drawing rectangles to the canvas			
.clearRect()	.clearRect()	R_GRAPHICS_Clear	✓
.fillRect()	.fillRect()	R_GRAPHICS_FillRect	✓
.strokeRect()	-	-	
10 Drawing text to the canvas			
	-	-	
11 Drawing images to the canvas			
.drawImage(dx, dy)	.drawImage()	R_GRAPHICS_DrawImage	✓
.drawImage(dx, dy, dw, dh)	.drawImage()	R_GRAPHICS_DrawImageResized	✓
.drawImage(sx, sy, sw, sh, dx, dy, dw, dh)	.drawImage()	R_GRAPHICS_DrawImageChild	✓
12 Pixel manipulation			
.createImageData(Width, Height)	.createImageData()	R_GRAPHICS_IMAGE_-InitR8G8B8A8	–
.createImageData(ImageData)	.createImageData()	R_GRAPHICS_IMAGE_-InitSameSizeR8G8B8A8	–
ImageData.width	ImageData.width	R_GRAPHICS_IMAGE_GetProperties	–
ImageData.height	ImageData.height	R_GRAPHICS_IMAGE_GetProperties	–
ImageData.data	ImageData.data	R_GRAPHICS_IMAGE_GetProperties	–
.getImageData()	.getImageData()	R_GRAPHICS_IMAGE_-InitCopyFrameBufferR8G8B8A8	x
.putImageData()	.putImageData()	R_GRAPHICS_DrawImage	x
		R_GRAPHICS_DrawImageChild	

13 Compositing			
.globalAlpha	.globalAlpha	R_GRAPHICS_SetGlobalAlpha	✓
.globalCompositeOperation	.globalCompositeOperation	R_GRAPHICS_-SetGlobalCompositeOperation	✓
14 Shadows			
	-	-	

6.11.2 Identifying Image Format

The first few bytes of the graphics_image_t-type structure to be specified for the argument of the R_GRAPHICS_DrawImage (section 6.9.1.29), R_GRAPHICS_DrawImageResized (section 6.9.1.30), or R_GRAPHICS_DrawImageChild (section 6.9.1.31) function are used to identify image formats. Therefore, JPEG file data or PNG file data can be specified directly for the graphics_image_t-type argument of the R_GRAPHICS_DrawImage function.

Image Format	First Few Bytes	Note
JPEG	0xFF 0xD8	SOI segment
PNG	0x89 0x50 0x4E 0x47	PNG header
Raw	Others	graphics_image_t-type structure + Raw data

6.11.3 Changing RGAH and RGAS

RZ/A1H,M can use RGA using hardware acceleration version (RGAH) and RGA using no hardware acceleration version (RGAS). But RZ/A1L can use RGA using no hardware acceleration version (RGAS) only.

Initialize example and Video Display Controller 5 (VDC5) driver are not same for RZ/A1H as for RZ/A1L. Get the example for another target chip.

The way of changing RGAH and RGAS in RGA is the following steps.

1. Change enabled macro "IS_RGAH_ARM_VERSION" or "IS_RGAS_ARM_VERSION" in "RGA_Sample\armcc\common\src\drivers\RGA\inc\RGA_Config.h".
2. In using RGAH, link RGA (ForRGAH), RGAH and PNG. jpeg is not linked. In using RGAS, link RGA (ForRGAS), jpeg and PNG. RGAH is not linked. See Table 2-2 Library files.
3. Convert the alignment of 1 line of drawing raw image to the specification of RGAH or RGAS. See "RGA_SOURCE_IMAGE_STRIDE_ALIGNMENT" constant value. In RGA example, run the batch file *RGAH.bat or *RGAS.bat in the following folder.
   RGA_Sample\armcc\common\src\samples\RGA\Sample_Common\Images
   BinaryImageMakeRGAH.bat
   BinaryImageMakeRGAS.bat

6.11.4 Flush Mode

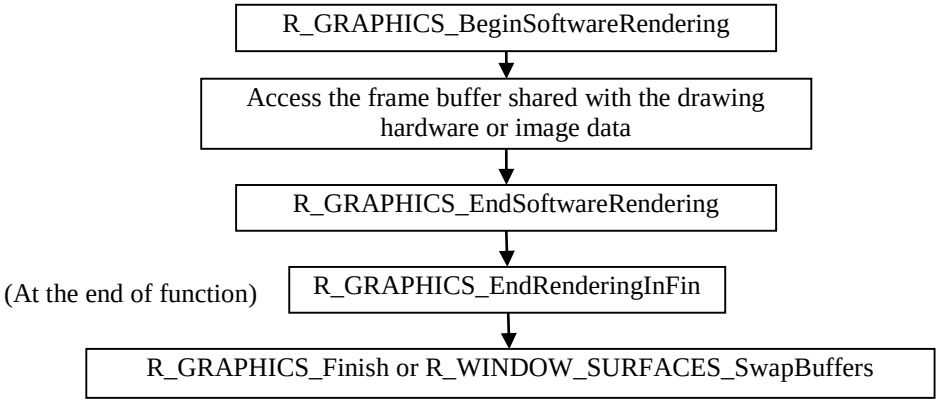
In auto-flush mode, drawn content can be displayed only by calling R_GRAPHICS_Finish or R_WINDOW_SURFACES_SwapBuffers. In fast manual flush mode, however, cache flush operation is required.

Auto-flush mode is enabled by default. In this mode, you need not pay attention to the description of this section.

To enable fast manual flush mode, set graphics_config_t::is_fast_manual_flush to "true". Since unnecessary flush operation is not performed in the graphics library in this mode, the processing may be performed quickly.

To perform the flush operation efficiently, enclose the processing that accesses the frame buffer shared with the drawing hardware or image data by R_GRAPHICS_BeginSoftwareRendering (section 6.9.1.48) to R_GRAPPHICS_EndSoftwareRendering (section 6.9.1.49).

When the application defined buffer was drawn by software, call "R_GRAPHICS_Finish" in similar to drawing by RGA.



The following describes specific examples of processing that must be enclosed.

- When directly reading or writing the frame buffer from outside the graphics library. However, enclosing this processing is not necessary when accessing the frame buffer from a non-cache area.
- When reading or writing image data specified for R_GRAPHICS_DrawImage or image data obtained from R_GRAPHICS_IMAGE_GetAddress. However, enclosing this processing is not necessary when reading or writing image data provided in the ROM.
- When this read or write operation is performed in the API that reads and writes files

When these functions are called, frame buffer data or image data is flushed from the CPU cache to the physical memory or hardware drawing is completed as needed.

If you do not want to use these functions, call R_GRAPHICS_Finish before accessing the frame buffer shared with the drawing hardware or image data. After the frame buffer or the image data is accessed, write back the CPU cache.

Sample

```
  e= R_GRAPHICS_BeginSoftwareRendering( graphics ); IF(e){goto fin;}

  /* Access to frame buffer directly */
  IF ( error ) { e=ERROR_CODE; goto fin; }

  e= R_GRAPHICS_EndSoftwareRendering( graphics ); IF(e){goto fin;}

fin:
  e= R_GRAPHICS_EndRenderingInFin( graphics, e );
```

6.11.5 Using SDRAM

For RZ/A1H, SDRAM is out of PL310 which has L2 cache. If the target frame buffer or the source image is in SDRAM, it is necessary to set and operate related with PL310.

6.11.5.1 Setting of enabling L2 cache

If "PL310_Enable" function was not called, write like the following code which calls "PL310_Enable" function for enabling L2 cache.

```
#include  "core_ca9_io.h"  /* for pl310.h */
#include  "pl310.h"  /* L2 cache */


    /* If present, you may also need to Invalidate and Enable L2 cache here
*/
    l2_id = PL310_GetID();
    if (l2_id)
    {
        PL310_InvAllByWay();
        PL310_Enable();
    }
```

6.11.5.2 Setting of NS bit in MMU of CPU

Set 1 at NS bit in MMU (TTB) of SDRAM area, if NS bit was set 0.

6.11.5.3 Setting of the cache attribute output from bus masters (AWCACHE, ARCACHE)

If memory map was changed, modify define of "R_OSPL_AXI_Get2ndCacheAttribute" function and "R_OSPL_MEMORY_GetLevelOfFlush" function in OSPL porting layer. RGA changes the cache attribute setting (AWCACHE, ARCACHE in the register of RZ/A1H) depend on whether bus masters Renesas Graphics Processor for OpenVG™ (R-GPVG2) and JPEG Codec Unit (JCU)) access On-Chip RAM or SDRAM. RGA gets the value of the cache attribute from above porting functions.

6.11.5.4 Changing calling functions for fast manual flush mode

In RZ/A1H, if a frame buffer or a source image was in SDRAM and all bus master in RZ/A1H access to SDRAM using L2 cache, it does not have to flush L2 cache. It is recommended this setting by set to MMU and define "R_OSPL_AXI_Get2ndCacheAttribute" function and "R_OSPL_MEMORY_GetLevelOfFlush" function. If MMU setting was needed to flush L2 cache and fast manual flush mode (6.11.4) was enabled, flush L2 cache by changing the code calling "R_GRAPHICS_BeginSoftwareRendering2" function or "R_GRAPHICS_BeginSoftwareRenderingA" function instead of "R_GRAPHICS_BeginSoftwareRendering" function.

6.11.6 Layer structure of sample screen controller

The following table describes the correspondence relation between the layer structure of window_surfaces_t (6.6.5.5) and planes of Video Display Controller 5 (VDC5).

Table 6-6 Layer structure of sample screen controller for RZ/A1H, M

id of layer_attributes_t (6.6.5.7)	The plane name of VDC5
(Not used)	Graphics (3)
1	Graphics (2) If " window_surfaces_config_t::pixel_format = PIXEL_FORMAT_YCbCr422", the pixel format becomes RGB565.
0	Graphics (1)
-1 (Background color)	Graphics (0)

Table 6-7 Layer structure of sample screen controller for RZ/A1L (pixel_format≠PIXEL_FORMAT_YCbCr422)

id of layer_attributes_t (6.6.5.7)	The plane name of VDC5
1	Graphics (3)
0	Graphics (2)
-1 (Background color)	Graphics (0)

Table 6-8 Layer structure of sample screen controller for RZ/A1L (pixel_format=PIXEL_FORMAT_YCbCr422)

id of layer_attributes_t (6.6.5.7)	The plane name of VDC5
(Not used)	Graphics (3)
1	Graphics (2) RG565.
0	Graphics (0) YCbCr422

6.11.7 Flagged Structure Parameters

Flags member variables in the structure are used as a bit field. When the bit is 1, the corresponding member variable is enabled in this coding pattern. When the bit is 0, it is treated that the default value is set or not change for the member variable as omitted. Updating the version allows binary compatibility even when structure members increase.

```
FuncA_ConfigClass  config;

config.Flags  = F_FuncA_Param1 | F_FuncA_Param2;
config.Param1 = 10;
config.Param2 = 2;
FuncA( &config );
```

The default value is config.Param3 because "Flags |= F_FuncA_Param3" does not exist.

6.11.8 Defaultable Flags

Defaultable flags are the type that can select three options: setting to ON, setting to OFF, and no setting (unchanged) for each array element when setting logic-type array variables. Symbols for setting to ON and setting to OFF are defined. When these symbols are not used, settings remain unchanged.

The variable type that stores defaultable flags is a 32-bit integer type. These flags are defined as a flag that sets the lower 16 bits to ON and a flag that sets the upper 16 bits to OFF. Components of the upper 16 bits are the same as those of the lower 16 bits.

```
typedef BitField  DefaultableFlagsType;  /* Flags of DefaultableFlagType */
enum DefaultableFlagType {
    /* Set to "ON" */
    ENABLE_SAMPLE_FLAG_A     = 0x0001,
    ENABLE_SAMPLE_FLAG_B     = 0x0002,
    ENABLE_SAMPLE_FLAG_C     = 0x0004,

    /* Set to "OFF" */
    DISABLE_SAMPLE_FLAG_A    = ENABLE_SAMPLE_FLAG_A << 16,
    DISABLE_SAMPLE_FLAG_B    = ENABLE_SAMPLE_FLAG_B << 16,
    DISABLE_SAMPLE_FLAG_C    = ENABLE_SAMPLE_FLAG_C << 16,
};
```

When the argument of the function to set flags (SampleClass_setDefaultableFlags below) is a defaultable flag, only flags to be set to "ON" or "OFF" are connected by "|". Settings remain unchanged for flags that are not set to "ON" or "OFF." For initialization functions, default values are used.

```
errnum_t  main()
{
    DefaultableFlagsType  flags;

    e= SampleClass_setDefaultableFlags( object,
        ENABLE_SAMPLE_FLAG_A | DISABLE_SAMPLE_FLAG_B ); IF(e)goto fin;
        /* SAMPLE_FLAG_C is not modified. */

    e= SampleClass_getDefaultableFlags( object, &flags ); IF(e)goto fin;
    if ( flags & ENABLE_SAMPLE_FLAG_A ) { ... }
    if ( flags & DISABLE_SAMPLE_FLAG_B ) { ... }
    if ( flags & ENABLE_SAMPLE_FLAG_C ) { ... }
}

errnum_t  SampleClass_setDefaultableFlags( SampleClass* self,
DefaultableFlagsType Flags )
{
    self->Flags = self->Flags | ( Flags & 0x0000FFFF );
    self->Flags = self->Flags & ~( Flags >> 16 );
}

errnum_t  SampleClass_getDefaultableFlags( SampleClass* self,
DefaultableFlagsType* out_Flags )
{
    BitField  flags = ( self->Flags & 0x0000FFFF );
    *out_Flags = flags | ~( flags << 16 );
}
```

When the argument of the function to acquire the current flag (SampleClass_getDefaultableFlags above) is a defaultable flag, the upper 16 bits of an acquirable value are an inverted value of the lower 16 bits. For this reason, both symbols for setting to ON and symbols for setting to OFF can be used for decision statements. Since the internal variable that retains the current flag is the internal specification, there is no problem with specifications in which upper 16 bits are disabled.

Inverted value of lower 16 bits (upper 16 bits)	Current flag (lower 16 bits)

6.11.9 Function to Initialize Internal Variables with Constants (*_initConst Function)

In the C language class that includes the finalizing function (*_finalize function), the *_initConst function must be called first (before calling the *_initialize function). This is to prevent any exception from occurring even if the *_finalize function is called when an error occurs from another object before initialization. Before calling the function that may cause an error first among functions, call the *_initConst function first at a time for all objects that only exist in functions (i.e. created and deleted in functions).

Even if the *_initConst function is called, many functions (methods) are not made available until the *_initialize function is called.

In the case of the C++ language API of this library, the *_initConst function responds to the constructor. The *_initialize function responds to object creation functions (such as R_RGA_New_Canvas2D_ContextClass function). The *_finalize function responds to the destroy member function. For example, when an error occurs before an object creation function is called after the constructor was called by the variable declaration, no exception occurs even when the destroy member function is called.

6.11.10 Finalize (*_Finalize Function)

The function that last name of is "_Finalize" does the operation of finalize. Finalize is close operation of file, the operation in the destructor of C++ language, the operation called from finally block of Java language and others.

*_Finalize		
Outline	Does finalize operation.	
Declaration	errnum_t *_Finalize(type* self, errnum_t e);	
Argument	type* self	Target object to finalize
	errnum_t e	Error code raised from before. There is different whether or not to do the rollback operation whether or not error was raised depending on a function's specification.
Return value	Error code or e 0 = successful and e = 0 If argument e is not 0, returns the value of e. If argument e is 0, returns an error code of this finalize operation.	

Pass error code raised from before to argument "e" like the following code.

```
errnum_t  Func()
{
    errnum_t  e;

    CLASS_InitConst( &sample );
    e= CLASS_Initialize( &sample ); IF(e){goto fin;}

    e=0;
fin:
    e= CLASS_Finalize( &sample, e );
    return  e;
}
```

If the finalize operation was success, the target object becomes Uninitialized state.

If error was raised in the finalize operation, the function returns error code not 0 and may become the following state.

State after finalizing	Description and expected behavior
Uninitialized	The target object can be deleted. In this case, all internal objects are finalized, even if errors were raised internally and the program can be resumed from error state. Uninitialized internal objects are deleted. Internal objects which returned to a previous state are not deleted until resource management object deletes it or the system was reset. Otherwise, the object may keep lock state. A callback function related internal object which returned to a previous state may be called. Notify the error information to the end user or make logs after finalize function.
Reset	Calls "R_OSPL_RaiseUnrecoverable" function from the finalize function. "R_OSPL_RaiseUnrecoverable" function resets the system or exits the process. In this case, finalize function is not returned.

(State before finalizing)	The object which returned to previous state can not be deleted. Call finalize operation which usually returns to previous state from out of finally block (fin: in above code) and retry the finalize operation. If the finalize operation is called from finally block, the object will not be deleted. Adjust anything before calling the finalize function. An object which integrates not deleted object sometimes becomes to Uninitialized state. In this case, see the description of "Uninitialized" in this table.

6.11.11 Compatibility between C++ Language and JavaScript Object

JavaScript codes using the Canvas 2D object can be operated as they are by using the C++ language class provided by this library. The following shows codes using variables of the class provided by this library, which are the same description of codes using variables that reference the JavaScript object.

- When assignment operation is performed, one object can be accessed from two variables. Therefore, an operation is performed as if the value of the pointer that indicates the C++ language object is copied.

```
object_1_reference = object_1;
```

- When accessing a member, describe a period instead of the "->" operator (hyphen + inequality sign).

```
object_1.attr = 1;
```

- Since the destructor is not automatically activated as in the case of JavaScript, explicitly call the destroy member function to delete objects. This corresponds delete operator of C++ language.

```
object_1.destroy();
```

The compatibility between the C++ language provided by this library and JavaScript applies only to codes for objects. In the C++ language, not all JavaScript codes operate as they are (such as required variable declaration).

When an error occurs, the return value of "R_OSPL_GetErrNum" function is not 0. No exception occurs. Call the "R_OSPL_CLEAR_ERROR" function when returning from the error. Even if a method of an error object is called, no internal processing is performed. However, "save" and "restore" methods do operations in the error state.

Sample code of error check code:

```
if ( R_OSPL_GetErrNum() != 0 ) { ... }
```

Sample code of error clear code:

```
R_OSPL_CLEAR_ERROR();
```

If an error was raised in the function creating an object, "R_OSPL_GetErrNum" function returns not 0 and undefined object is returned from the creating function. Whether the return value is the undefined object or not can be determined by == operator of the object handle variable.

7. Tools

7.1 Image Format Conversion by ImagePackager

The ImagePackager packs multiple image files into a single binary file (for the target board) or a source file (for the target board and PC). When image files are packed, they can also be converted to the raw format of any pixel formats. The integrated file is used as a source of binary data or C language data. In an XML file, specify image files to be packed. File names and extension name are not ignored case-insensitive comparison.

ImagePackager is one of commands in "RGA_Tools.vbs". It calls vbs files and exe files internally.

7.1.1 Operational Procedure

1. Make .image.xml file ().
2. Double click RGA_Tools.vbs file at armcc\common\src\samples\RGA, select "ImagePackager" command in opened window, specify .image.xml file. This makes header file and binary file or C language data source.

```
------------------------------------------------------------------------------
RGA Tools - Copyright(c) 2012-2015 Renesas Electronics Corporation
 1. Convert image format [RunImagePackager]
 2. Search error information [SearchErrorInformation]
Number or command >1
------------------------------------------------------------------------------
((( [RunImagePackager_sth] )))
Renesas Image Packager - Copyright(c) 2012-2015 Renesas Electronics
Corporation
Make one binary file from many image files and other files.
Enter only : Open sample folder.
Setting file(ImagePackagerConfig.xml) >
```

3. Write #include directive with a generated header file and pass a symbol of image defined in the header file to "R_GRAPHICS_DrawImage" function or other functions.
4. If binary file was generated, write it at the address written in .image.xml file and start drawing operations. The address is written at /ImagePackager/OutputBinary/OutputHeader/@address in .image.xml file or generated header file.

ImagePackager can be started by the following command in the command prompt.

```
>cd  armcc\common\src\samples\RGA\Sample_Common\Images
>cscript  //nologo  ..\..\RGA_Tools.vbs  RunImagePackager
BinaryImageConfig.image.xml
```

7.1.2 List of files

File	Description
RGA_Tools.vbs	Script file which contains ImagePackager
*.image.xml	Setting file of ImagePackager
(*.bmp, *.jpg, *.png)	Input image file
(Output: binary file)	A file which contains images and files to download to the target board
(Output: source file)	A source file which contains image data and file data to put in the program

7.1.3 Sample

BinaryImageConfig.image.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ImagePackager>


<OutputBinary path="BinaryImage_RZ_A1H.c" language="C"
symbol="g_RGA_Sample_BinaryImage"
   source_template="${ImagePackagerLib}\SourceTemplate.xml#default"
   super_class="${ImagePackagerLib}\SuperClass.xml#default">
<OutputHeader path="BinaryImage_RZ_A1H.h"
include_define="BINARYIMAGE_RZ_A1H_H"/>
</OutputBinary>


<InputFiles>
<File path="BinaryHeader.txt" type="char[]" symbol="g_BinaryHeader"/>
<Image path="picture.bmp" type="graphics_image_t*" symbol="g_Picture_bmp"
output_format="RGB565"/>
<Image path="smile32.bmp" type="graphics_image_t*" symbol="g_Smile_bmp"
output_format="ARGB4444"/>
<File path="JPEG.jpg" type="graphics_image_t*" symbol="g_JPEG_jpg"/>
</InputFiles>


</ImagePackager>
```

The folder that contains the XML file is the reference of relative paths.

A folder path or wild card can be specified for Image/@path.

Providing a fixed-value header at the top allows checking whether binary data is contained or not.

7.1.4 Types of Output Binary File (Language)

- Binary format (external reference symbol of C language) + C language header
- Binary format or S-record format (direct addressing such as in flush) + C language header
- C language source + C language header

Specified for /ImagePackager/OutputBinary/@language and /ImagePackager/OutputHeader/@address

7.1.5 File Formats in the Output Binary File

Format	Description
Offset table	The array of 4byte integer type in the first of the binary file that indicates the offset value to image data and binary data.
Raw-format image	Specified for /ImagePackager/InputFiles/Image/@output_format "ARGB8888", "XRGB8888"(X component is 0x00), "RGB565", "ARGB1555", "ARGB4444", "YUV422", "CLUT8","CLUT4","CLUT1","A8","A4","A1" See following "The format of Raw-format image"
Binary format	When expanding JPEG data on the target board, the file is stored as it is. Specified for /ImagePackager/InputFiles/File

The format of Raw-format image:

Offset	Size	Description
0x00	4byte	The bit flags described the type of image [bit0] 1fixed [bit1] (reserved) [bit2] 1=Premultiplied alpha, 0=not Premultiplied alpha [bit3-bit15] (reserved) [bit16-bit31] 0 This endian is depended on the setting of "@endian".
0x04	4byte	The offset value to the image data from the first of Raw-format image. This endian is depended on the setting of "@endian".
0x08	4byte	(reserved)
0x0C	2byte	The image width (pixels) This endian is depended on the setting of "@endian".
0x0E	2byte	The image height (pixels) This endian is depended on the setting of "@endian".
0x10	1byte	The pixel format 0=RGB565, 2=ARGB1555, 3=ARGB4444, 6=CLUT8, 7=CLUT4, 8=CLUT1, 9=XRGB8888, 10=ARGB8888, 11=YUV422, 20=A8, 21=A4, 22=A1
0x11	7byte	(reserved)
*		Image data If the pixel format was RGB components, this endian is depended on the setting of "@endian". If the pixel format was YUV components, this endian is not depended on. The offset value of this image data is depended on the setting of "@raw_image_alignment". The byte count par one line (stride) is depended on the setting of "@raw_stride_alignment" and so on. If this image data had CLUT (Color Look Up Table, Palette), there are image data after CLUT. The elements of CLUT are ARGB8888 format depended on the endian by "@endian" setting. The count of elements are 256 (CLUT8), 16 (CLUT4) or 2(CLUT1).

7.1.6 Input Formats

- BMP format: 32 bits or 24 bits (256, 16, or 2 colors for CLUT)
- PNG format: Supports data with alpha
- JPEG format

See the description on @path of "/ImagePackager/InputFiles/Image".

7.1.7 Parameters That Can Be Described in BinaryImageConfig.image.xml

/ImagePackager/OutputBinary: One or more parameters (Two or more parameters are used to output both "Binary" @language and "C" @language.)

@path	Path of the binary file output destination (Essential)
@symbol	C language's external reference symbol (global variable name) that supports all binary files (Essential) When outputting multiple binary files, do not change this parameter in each binary file. When /ImagePackager/OutputHeader/@address is set, this parameter is a macro name having the address value.
@language	Programming language of output binary data (Optional) "Binary" and "C", "SRec"(S-record) can be set. "Binary" is set by default.
@section	Specified section is embedded in the output source. Example: section = "_BinaryImage" --- C_BinaryImage section is created. This is filled at ${Section} in /ImagePackager/SourceTemplate.

@endian	Endian (Optional) "LittleEndian" or "BigEndian" can be selected. @endian pointed from @super_class is set by default.
@raw_image_alignment	Alignment (bytes) of the starting address of raw-format image data (Optional) Only 2^n can be set. @raw_image_alignment pointed from @super_class is set by default. When outputting multiple binary files, do not change this parameter in each binary file. The raw-format header is not aligned by this setting.

@raw_image_alignment_symbol	The #define symbol name containing the @raw_image_alignment value (Optional)
	The #define symbol name is output to the header file to be generated. Compile the program using binary files by using the header.
	`Example: raw_image_alignment_symbol= "GRAPHICS_RAW_IMAGE_ALIGNMENT"`
	The code filled in the header file:
	`#define RAW_IMAGE_ALIGNMENT 32`
	
	The #define sentence of the symbol name is not output by default. When outputting multiple binary files, do not change this parameter in each binary file.

@raw_stride_alignment	Alignment of the number of bytes up to the line immediately below in an image to be output (Optional). The number of bytes is carried to a multiple of the value specified for (value). @raw_stride_alignment pointed from @super_class is set by default. `Example: raw_stride_alignment = 32`
@raw_stride_alignment_4	Pixel format of output image in which the number of bytes up to the line immediately below is 4. Two or more CSV formats can be specified. (Optional) This setting takes precedence over the @raw_stride_alignment setting. `Example: raw_stride_alignment_4 = YUV422`
@table_format	The format of the table. (Optional) ● "Offset": Fills not only the body but also the offset to the body in the binary file, because re-compile does not have to be done, even if the size of the image or the file was changed. ● "Embed": Fills images and files normally. Default value is the following value. ● If @language="Binary", @table_format="Offset" ● If @language="C", @table_format="Embed"
@source_template	ID of the template of the output source file in the case of @language="C". (Optional) Set the value of "/ImagePackager/SourceTemplate/@id". It is able to set the reference to the other XML file. ${ImagePackagerLib} is replaced to the folder path containing ImagePackagerLib.vbs `Example: source_template="${ImagePackagerLib}\SourceTemplate.xml#default"`
@super_class	ID of the super class attached to the binary file. (Optional) Set the value of "/ImagePackager/SuperClass/@id". It is able to set the reference to the other XML file. ${ImagePackagerLib} is replaced to the folder path containing ImagePackagerLib.vbs

```
Example:
super_class="${ImagePackagerLib}\SuperClass.xml#d
efault"
```

/ImagePackager/OutputBinary/OutputHeader: One or more elements

/ImagePackager/OutputHeader: Zero or one element

@path	Path of the header file output destination (Essential)
@include_define	The #define symbol name to prevent the header file from being included twice (Optional) Default name: "__BINARY_IMAGE__"
@address	Memory address to allocate binary files (Optional) Specify the memory address in the 0x00000000 format. This parameter is used when allocating addresses directly in the flash memory in addition to the program image to be linked. Since the set address is output to the header file to be created, compile the program that references binary data by using the header. If this parameter is omitted, the C language's external reference symbol (global variable name) specified for /ImagePackager/OutputBinary/@symbol is used.

/ImagePackager/InputFiles: Only one element

@base_folder	A path to be the reference of /ImagePackager/InputFiles/Image/@path (Optional) The reference of relative paths is the folder that contains the BinaryImageConfig.image.xml file. Default path: "."

/ImagePackager/InputFiles/Image: Zero, one or more elements

The data converted to Raw-format are embedded in the binary file.

@path	Path of image files to be input (Essential) As the reference of relative paths, the path of the /ImagePackager/InputFiles/@base_folder folder and the sub-folder are also input when a wild card is specified. In the case of "png" or "jpg" extension, incompressibly expanded files are output to binary files. When expanding a file on the target board, specify JPEG file for /ImagePackager/InputFiles/File. For images with no alpha channel or images whose all alpha values are 0xFF, information that shows that alpha blending is not required is embedded in the output data. This information may allow fast drawing.
@type	Type of symbols described in the header file (Essential) @symbol type. Specify graphics_image_t*.
@symbol	A symbol described in the header file (Essential) The #define symbol in the global scope. @type type When "${...}" is used, the symbol is replaced with a file name. `Example: g_${BaseName}_${Extension}_${Format} -> g_Sample_jpg_ARGB8888`
@output_format	Format of files to be output (Essential) If input file was 24bit/32bit Window bitmap file, JPEG file, PNG file (with alpha/without alpha), values "ARGB8888", "XRGB8888"(X component is 0x00), "RGB565", "ARGB1555", "ARGB4444", "YUV422", "A8", "A4" and "A1" can be specified. When "CLUT8" is specified, input a 256-color Windows bitmap file. When "CLUT4" is specified, input a 16-color Windows bitmap file. When "CLUT1" is specified, input a monochrome Windows bitmap file.

The following table describes alpha component of output image. If input image was 256, 16 colors or monochrome Windows bitmap file, see the input color as CLUT referenced color at the following table. Notice: PNG format created by Paint Brush for Windows 7 always has alpha component.

Input Image	Output Image	Alpha component of output
With A	Every format	A component of the input image
Without A	ARGB	0xFF
	A component only	Y component (luminance) converted input image from RGB to YCbCr

Two or more values can be specified in the CSV format. In that case, however, include "${Format}" in @symbol. "${Format}" will be replaced to the name of pixel format.

@premultiplied_alpha — This parameter shows whether to multiply RGB components by the alpha component. (Optional)

- "no": Not multiplied (default)
- "yes": RGB components are converted to multiplied values before they are output. This choice is available only when the alpha component is not contained in the drawing destination.
- "already_yes": The input image has already been multiplied. This choice is available only when the alpha component is not contained in the drawing destination.

/ImagePackager/InputFiles/File: Zero, one or more elements

Not converted data are embedded in the binary file.

@path	Path of files to be input (Essential) Reference of relative paths: /ImagePackager/InputFiles/@base_folder
@type	Type of symbols described in the header file (Optional) @symbol type. When the file content is an array, attach [] to the end of the type name. uint8_t[] is set by default.
@symbol	A symbol described in the header file (Essential) The #define symbol in the global scope. uint8_t[] type.
@alignment	The address alignment of the file. (Optional) @alignment pointed from /ImagePackager/OutputBinary/@super_class is set by default.

/ImagePackager/InputFiles/Var: Zero, one or more elements

The values written in XML file are embedded in the binary file.

@type	Type of the symbol written in the header file. (Essential) int32_t, uint32_t, int16_t, uint16_t, int8_t, uint8_t can be specified.
@symbol	The symbol name written in the header file. (Essential) This name is a name of global variable initialized the value embedded in the binary file.
@value	The value embedded in the binary file. (Essential) Integer or the following special format can be specified. ● Example of integer: "10", "-10", "0xFF"

- Special format "(new Image('file_path')).width" : Width of image named by file_path
- Special format "(new Image('file_path')).height" : Height of image named by file_path

"file_path" is the path of image file.

/ImagePackager/SourceTemplate/ : Zero, one or more elements

@id	ID of SourceTemplate tag. (Essential) This value is referred from /ImagePackager/OutputBinary/@source_template.
Source/text()	Template of source file. (Essential) The following tags can be filled in the template. ● ${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● ${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● ${Size} : Size of the binary file (bytes) ● ${BinaryData} : Binary data
SourceWithSection/text()	Template of source file, if a section was specified. (Optional) If SourceWithSection tag was not specified, the content specified by Source tag is used, even if a section was specified.
Header/text()	Template of header file. (Essential) The following tags can be filled in the template. ● ${include_define} : Macro name avoiding double include. The value of /ImagePackager/OutputBinary/OutputHeader/@include_define ● ${DeclareBinaryImageSymbol} : Declares of symbols in the binary data. The content of DeclareVariable/text() or DeclareAddress/text() and #define by @raw_image_alignment_symbol ● ${Variables} : #define list as variables ● ${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● ${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● ${Size} : Size of the binary file (bytes) ● ${StartAddress} : Start address of the binary data ● ${LastAddress} : Last address of the binary data
HeaderWithSection/text()	Template of header file, if a section was specified. (Optional) If HeaderWithSection tag was not specified, the content specified by Header tag is used, even if a section was specified.
DeclareVariable/text()	If @language="C", template filled at ${DeclareBinaryImageSymbol}. The following tags can be filled in the template. ● ${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● ${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● ${Size} : Size of the binary file (bytes)
DeclareAddress/text()	If @language="Binary", template filled at ${DeclareBinaryImageSymbol}. The following tags can be filled in the template. ● ${Section} : Section name. The value of /ImagePackager/OutputBinary/@section

- ${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol
- ${StartAddress} : Start address of the binary data
- ${LastAddress} : Last address of the binary data

/ImagePackager/SuperClass: Zero, one or more elements

@id	ID of SuperClass tag. (Essential) This value is referred from /ImagePackager/OutputBinary/@super_class.

/ImagePackager/SuperClass/OutputBinary : Zero or one

@endian	Default value of /ImagePackager/OutputBinary/@endian. This default value is "LittleEndian".
@raw_image_alignment	Default value of /ImagePackager/OutputBinary/@raw_image_alignment. This default value is 4.
@raw_stride_alignment	Default value of /ImagePackager/OutputBinary/@raw_stride_alignment. This default value is 1.
@raw_stride_alignment_4	Default value of /ImagePackager/OutputBinary/@raw_stride_alignment_4. This default value is "".

/ImagePackager/SuperClass/InputFiles/File : Zero, one or more elements

@path	Path of target file attached with SuperClass. Wildcard can be specified. (Essential) Example: path="*.jpg"
@alignment	Default value of /ImagePackager/InputFiles/File/@alignment. This default value is 4.

7.2 Searching for Error Information by SearchErrorInformation

When an error occurs in the debug-version library, the R_DEBUG_BREAK_IF_ERROR function shows the location raising an error. For RZ/A1, it output the following message to serial output. If target board did not have serial port, there is the way of getting error information by using LED and so on. (See following sentence).

printf output:

```
<ERROR error_ID="1" file="C:\folder\Library.c(2095)"/>
```

If you want to change to debug configuration, change the linking library from "lib\RGA\For<SubModule>\Release\RGA.a" to "lib\RGA\For<SubModule>\Debug\RGA.a", delete "R_OSPL_NDEBUG=1" in #define of the project settings.

When the SearchErrorInformation command is executed by double-clicking RGA_Tools.vbs, more advanced error information than the error code can be obtained from the file name and the line number in the library.

Window after RGA_Tools.vbs is double-clicked:

```
RGA Tools - Copyright(c) 2012-2015 Renesas Electronics Corporation
1. Image format conversion [RunImagePackager]
2. Error information search [SearchErrorInformation]
Number or command >2
-----------------------------------------------------------------
  ((( [SearchErrorInformation_sth] )))
Path or file name of error source file >Library.c
Error line number >2095
SoftFillRectangle_YUV422(): Rectangle.Left must be even
SoftFillRectangle_YUV422(): Rectangle.Left must be an even number.
```

If you want to break the location raising the error, call "R_OSPL_SET_BREAK_ERROR_ID" function at the first of the program. The argument is "error_ID" attribute of the above "ERROR" tag.

```
R_OSPL_SET_BREAK_ERROR_ID( 1 );
```

In RZ/A1 connected with ULINK2, after connecting the target (loading the program), open "src\drivers\ospl\porting\DebugBreak.c" file by drag and drop to DS-5, set a breakpoint by right click at the left of the line defined "R_DebugBreak" function [DS-5 breakpoints > Toggle Hardware Breakpoint].

When the program was re-started, CPU breaks at the location raising the error. In DS-5, caller function is shown in "Debug Control" view.

Even if the target board did not have serial port, you can get information from LED or GPIO connected with oscilloscope. The operational procedure is to call LED control function and waiting function (e.g. R_OSPL_Delay) from R_DebugBreak function and display the argument of R_DebugBreak function like Morse code.

Example:

- At start, turn on for 1 second and turn off for 1 second
- If bit was 1, turn on for 0.5 second and turn off for 0.5 second
- If bit was 0, turn on for 0.2 second and turn off for 0.8 second
- Display next binary digit by shift operation

7.3 Converting binary by ConvertBin

ConvertBin converts from binary file to C language array or S-record format (Motorola S-record).

Command	Description
BinToC	Converts from binary file to C language array. Variable name can be specified.
BinToSRec	Converts from binary file to S-record format. Comment, load address and execute address can be specified. Execute address is usually 0 for data binary.
SRecToBin	Converts from S-record format to binary file.

Window after RGA_Tools.vbs is double-clicked:

```
RGA Tools - Copyright(c) 2012-2015 Renesas Electronics Corporation
 1. Convert image format [RunImagePackager]
 2. Search error information [SearchErrorInformation]
 3. Convert to binary [ConvertBin]
Number or command >3
--------------------------------------------------------------------------
--
  ((( [ConvertBin] )))
1. Binary to C language [BinToC]
2. Binary to S-record format [BinToSRec]
3. S-record format to binary [SRecToBin]
Number or command >
```

8. Sample Codes

Sample codes are contained in "src\samples\RGA" in the DS-5 project.

See the attached document "RGA Tutorial" about the operational procedure.

9. Reference Documents

User's Manual: Hardware

RZ/A1 Group User's Manual (Hardware)

The latest version can be downloaded from the Renesas Electronics website.

R7S72100 RTK772100BC00000BR (GENMAI) User's Manual

The latest version can be downloaded from the Renesas Electronics website.

R7S72100 CPU (GENMAI) Optional Board RTK7721000B00000BR User's Manual

The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C

The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0

The latest version can be downloaded from the ARM website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

ARM Software Development Tools (ARM Compiler toolchain, ARM DS-5 etc) can be downloaded from the ARM website.

The latest version can be downloaded from the ARM website.

HTML Canvas 2D Context

W3C Candidate Recommendation 29 March 2012

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.

Revision History

		Description	
Rev.	Date	Page	Summary
1.03	Jul. 4, 2014	—	First edition issued
1.10	Dec, 16, 2014	—	Supported alpha only source image. Enlarged max size of source image.
1.20	Feb. 10, 2015	—	Supported for RZ/1L. Supported PNG. Added "GRAPHICS_COPY", "lock_object". Reduced porting work of RGPNCG.
2.00	Jul. 31, 2015	—	Second edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2015 Renesas Electronics Corporation. All rights reserved.

Colophon 5.0